Bank Marketing Effectiveness Prediction

using various Machine Learning Models

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Kothuri Guru Sai Keerthana | AP21110011578

Lakku Manohar Reddy | AP21110011581

Amancha Phalgun | AP21110011587

Botla Venkata Ramamohan Rao | AP21110011598

Under the Guidance of

Dr. Satya Pramod Jammy

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

May, 2024

Table of Contents

Abstract

This project explores the application of machine learning techniques to predict the effectiveness of bank marketing campaigns using a dataset from a Portuguese banking institution. With a focus on classification, the goal is to develop models that can accurately classify clients' responses to campaigns as positive or negative. The dataset contains various input variables such as age, job, marital status, education, and financial indicators.

Initial data analysis involved computing descriptive statistics and visualizations to understand the relationships between variables. Outliers were addressed using interquartile range, and missing values were imputed or features were eliminated if they contained over 50% null values. Insights from the analyses revealed patterns such as age group preferences, job categories, marital status, education levels, and loan statuses affecting subscription to term deposits.

The implemented algorithms, k-Means clustering, SVM, KNN, and Logistic Regression, underwent training and evaluation. Cross-validation techniques were employed to enhance model performance.

Our goal is to develop robust ML models that are accurate .To evaluate the performance of the models, we employ metrics such as accuracy, precision, recall, and F1-score.

The findings offer valuable insights into campaign effectiveness and demonstrate the potential of machine learning in optimizing marketing strategies. Though only a subset of algorithms was implemented, the study lays the groundwork for future analyses leveraging additional techniques to further enhance prediction accuracy.

Abbreviations

EDA	Exploratory Data Analysis
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
KNN	K Nearest Neighbours
SVM	Support Vector Machine

List of Figures

List of Equations

Confusion Matrix:

Figure 1 Confusion Matrix

Accuracy $= \dfrac{TP+TN}{TP+TN+FP+FN}$

Precision $= \dfrac{TP}{TP+FP}$

Recall $= \dfrac{TP}{TP+FN}$

F1 Score $= 2 \times \dfrac{Precision \times Recall}{Precision+Recall}$

KNN Distance Metrics:

1.Euclidean Distance : $\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d}(x_j - X_{i_j})^2}$

2.Manhattan Distance: $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$

3.Minkowski Distance: $d(x, y) = (\sum_{i=1}^{n} (x_i - y_i)^p)^{\frac{1}{p}}$

Logistic Regression:

1.Sigmoid Function: $\sigma = \frac{1}{(1+e^{-x})}$

2.Cost Function: $cost = -\frac{1}{m}\sum_{i=1}^{m}[y * log(a) + (1-y) * log(1-a)]$

3.Gradient Descent:

$$dW = \frac{\partial COST}{\partial W} = (A - Y) * X^T \text{ ...... shape (1 x n)}$$

$$dB = \frac{\partial COST}{\partial B} = (A - Y)$$

$$W = W - \alpha * dW^T$$

$$B = B - \alpha * dB$$

1. Introduction

Bank marketing campaigns play a crucial role in financial institutions' efforts to attract and retain customers. Understanding the factors that influence the effectiveness of these campaigns is essential for optimizing marketing strategies and maximizing returns on investment. In this context, machine learning techniques offer a promising avenue for predicting client responses to marketing initiatives. Leveraging a dataset provided by a Portuguese banking institution, this project delves into the predictive modeling of bank marketing campaign effectiveness. By analyzing a wide range of client demographics and financial indicators, the aim is to develop accurate classification models capable of discerning whether clients are likely to subscribe to term deposits based on campaign outreach.

The dataset comprises a rich array of input variables, including age, job type, marital status, education level, and financial status indicators such as balance and loan status. Initial exploratory data analysis reveals intriguing insights into the relationships between these variables and clients' propensity to subscribe to term deposits. From age group preferences to the influence of job categories and loan statuses, the data illuminates various factors that may impact campaign effectiveness. Such insights serve as a foundation for building robust predictive models that can inform targeted marketing strategies tailored to specific client demographics and financial profiles.

Addressing data preprocessing challenges, including missing values and outliers, is paramount to ensure the integrity and efficacy of the predictive modeling process. Through techniques such as imputation, feature elimination, and outlier treatment using the interquartile range, the dataset is refined for further analysis. Moreover, the presence of class imbalance, with a significantly higher number of clients not subscribing to term deposits compared to those who do, necessitates the adoption of oversampling techniques like Synthetic Minority Oversampling Technique (SMOTE) to mitigate bias and enhance model performance.

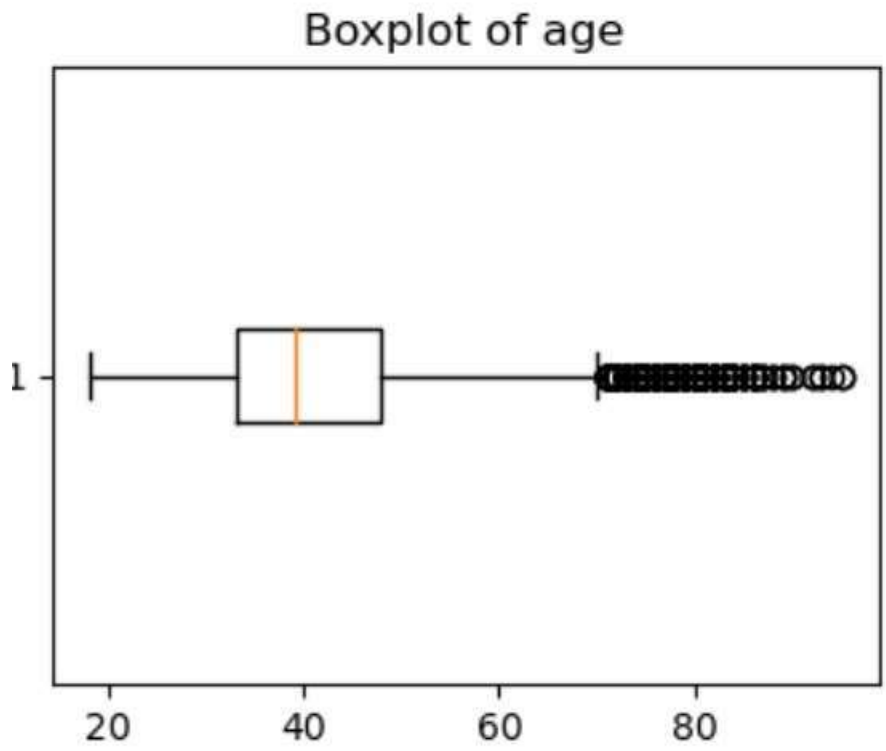
With a focus on implementing key machine learning algorithms, namely kMeans clustering, Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Logistic Regression, this project aims to provide a comprehensive analysis of bank marketing campaign effectiveness prediction. By leveraging the strengths of these algorithms and employing cross-validation techniques to refine model performance, the study endeavors to offer actionable insights for financial institutions seeking to optimize their marketing strategies and drive better campaign outcomes.

2.Dataset Description

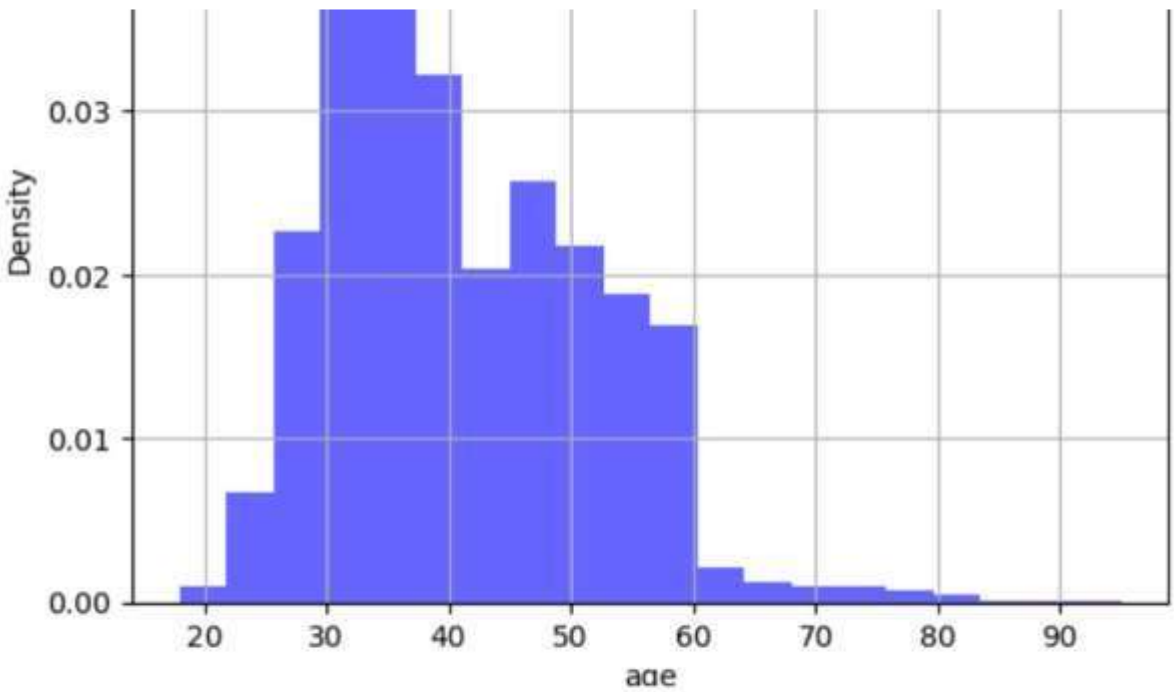
The dataset utilized in this project is obtained from a Portuguese banking institution and contains crucial information related to bank marketing campaigns. It comprises 45211 observations and 17 columns, each providing valuable insights into various aspects of client interactions and campaign outcomes. The dataset includes demographic details such as age, job type, marital status, and education level, along with financial indicators like account balance, loan status, and contact method. Additionally, temporal variables such as the day and month of contact, as well as campaign-specific metrics like call duration and the number of contacts, are incorporated into the dataset.
The data set contained details about bank marketing campaigns. Descriptive statistics were computed for each variable as part of the analysis, and visualizations were made to investigate the relationships between the various variables. We created a number of graphs, such as a distplot, count plot, bar plot, pair plot to gain insight from the dataset.

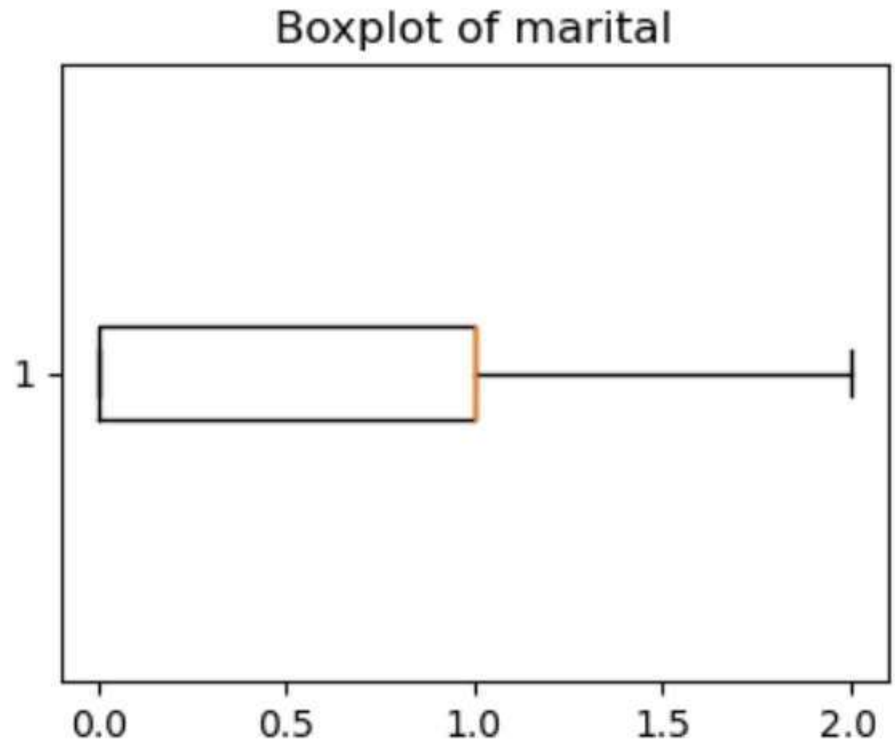
Categorical variables within the dataset, including job, marital status, education, contact method, and outcome of the previous marketing campaign, offer valuable insights into clients' socio-demographic backgrounds and previous interactions with the bank. These variables provide context for understanding client behavior and preferences, which are essential for predicting campaign effectiveness. Furthermore, numerical variables such as age, account balance, and call duration offer quantitative measures of clients' financial status and engagement with the marketing campaign. The target variable, denoted as 'y', indicates whether a client subscribed to a term deposit following the marketing campaign, facilitating the classification task of predicting campaign effectiveness.

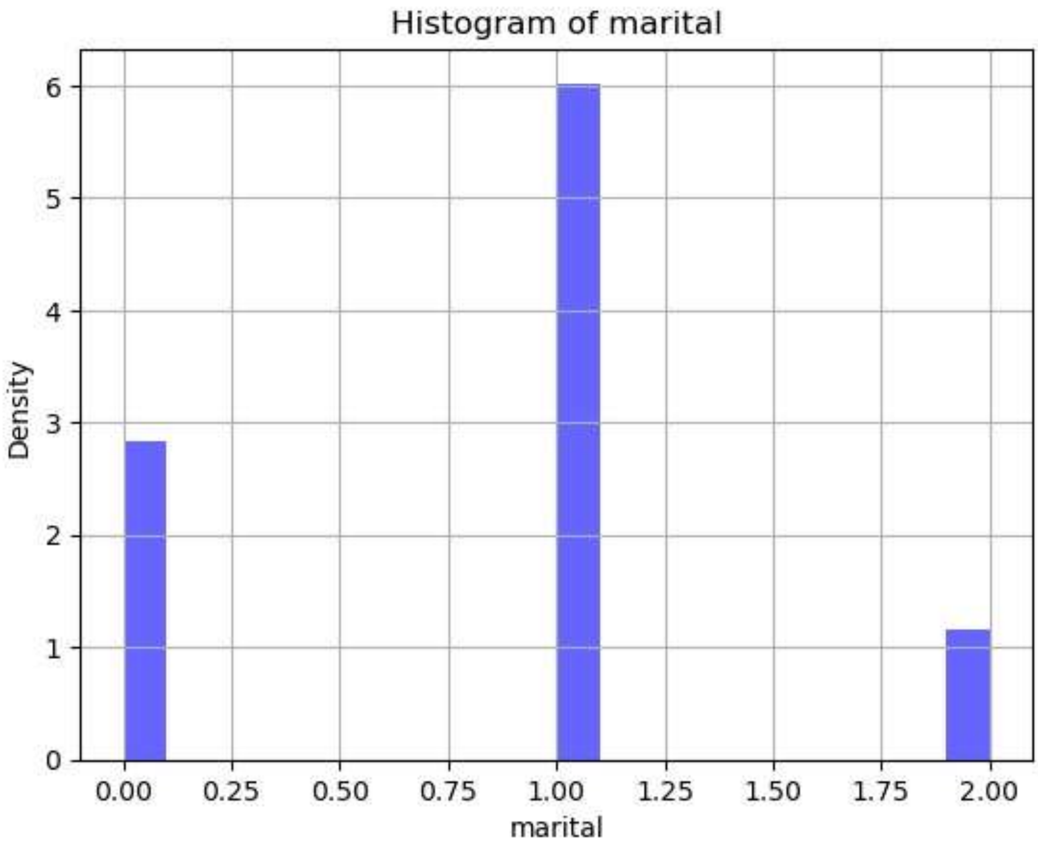
Data Preprocessing and handling Outliers:

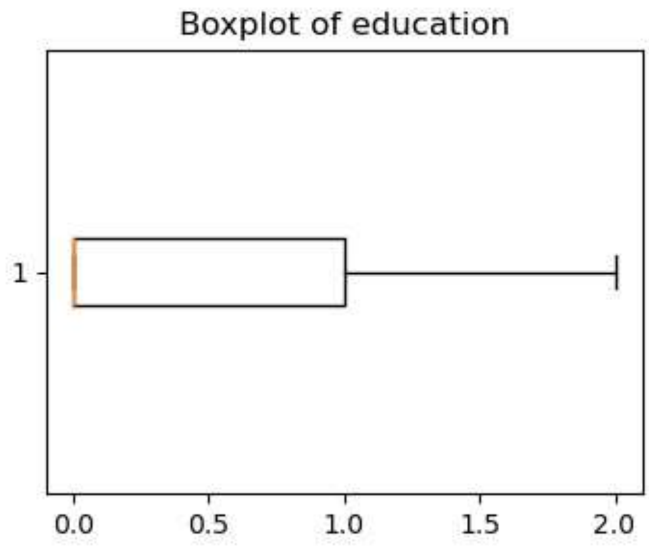


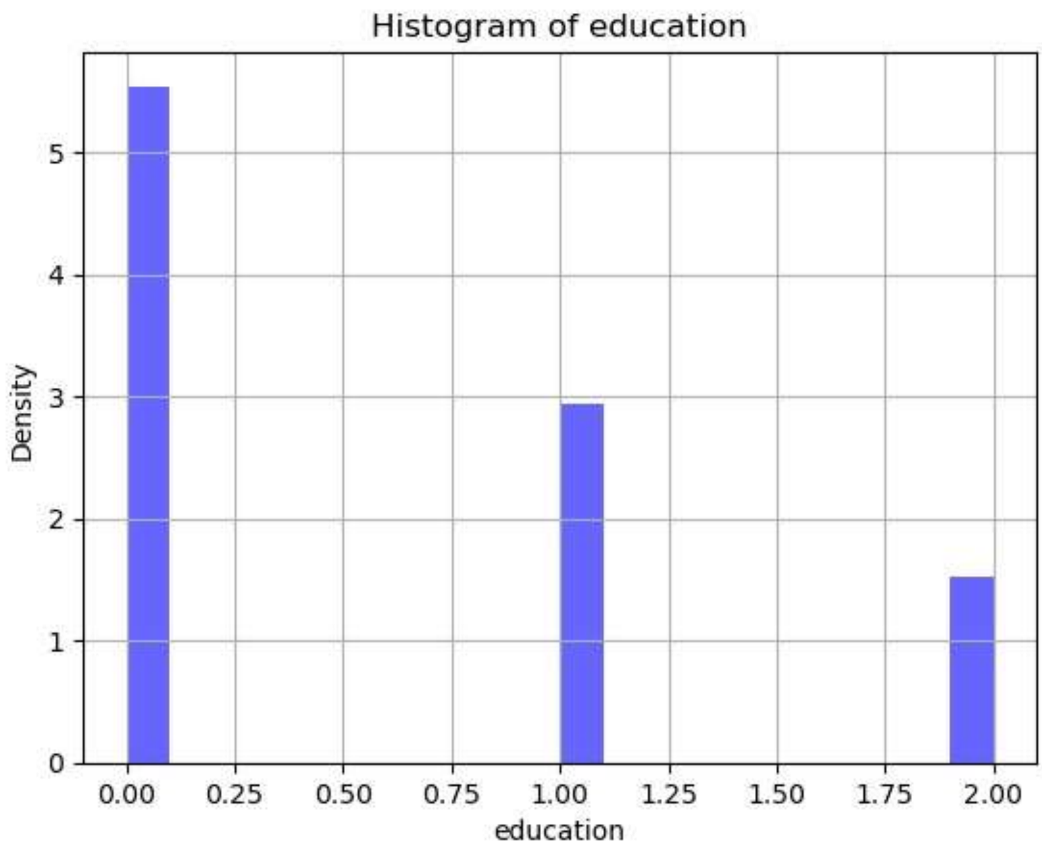
Histogram of Age:

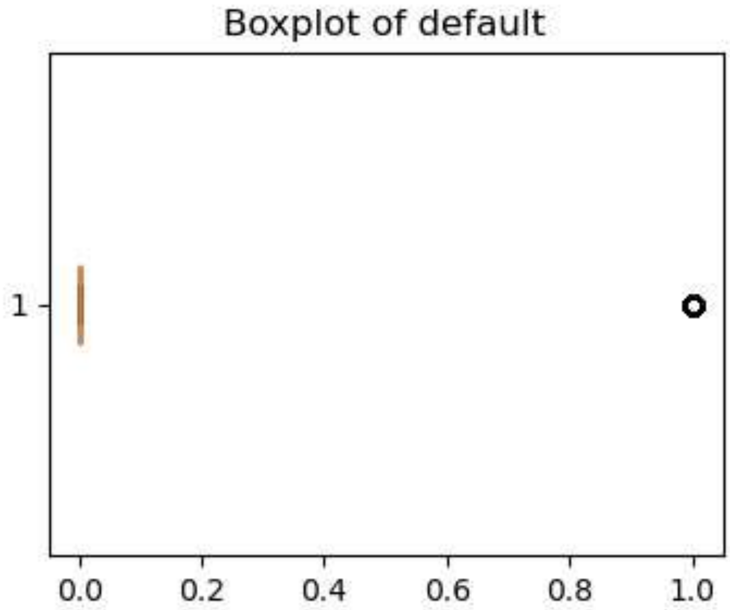


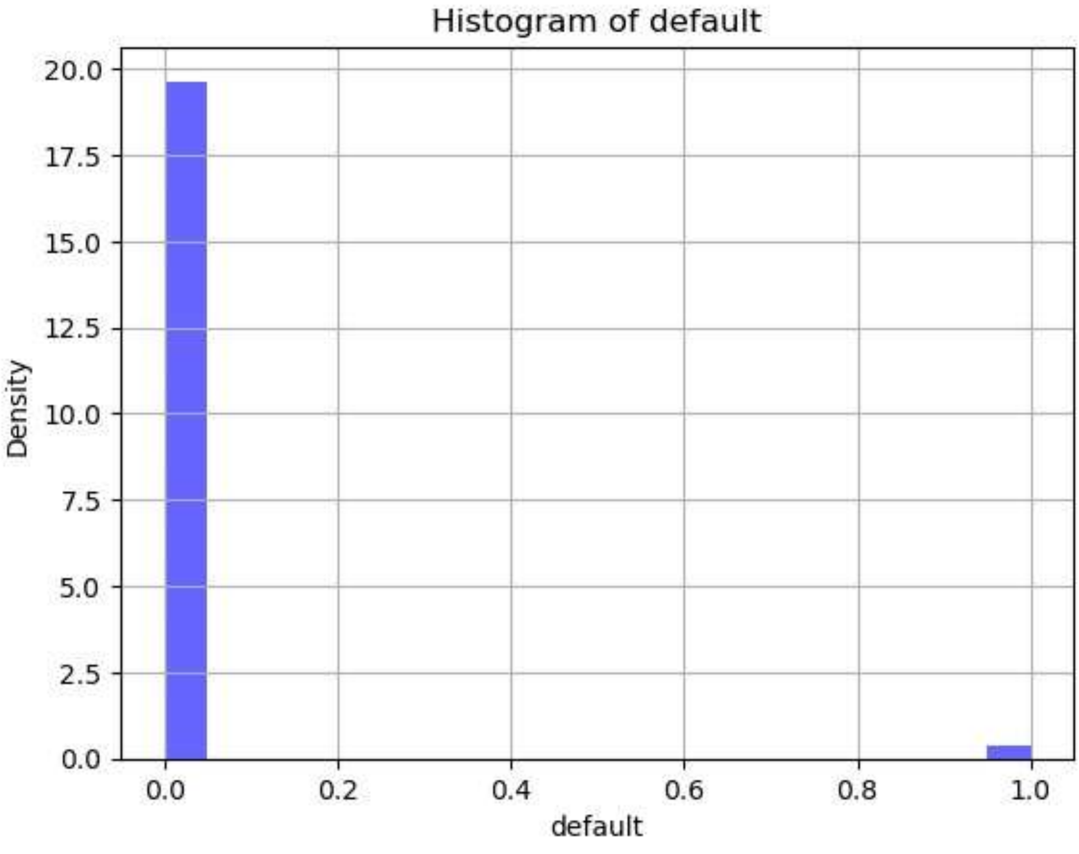

Boxplot of marital
1
0.0
0.5
1.0
1.5
2.0

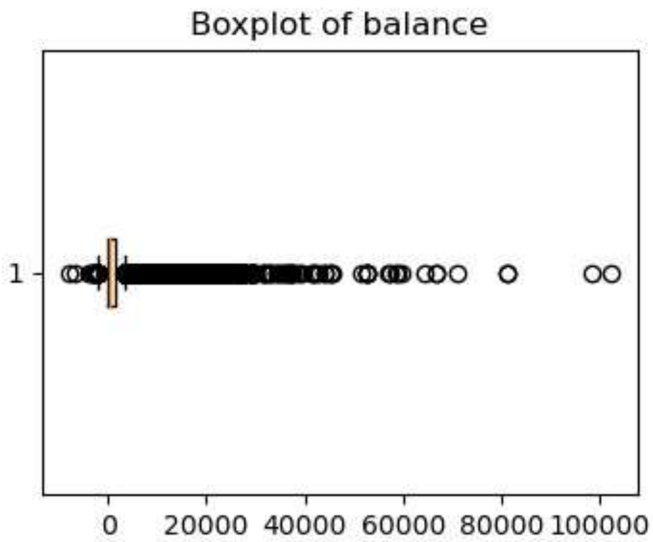

Histogram of marital
Density
0
1
2
3
4
5
6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
marital

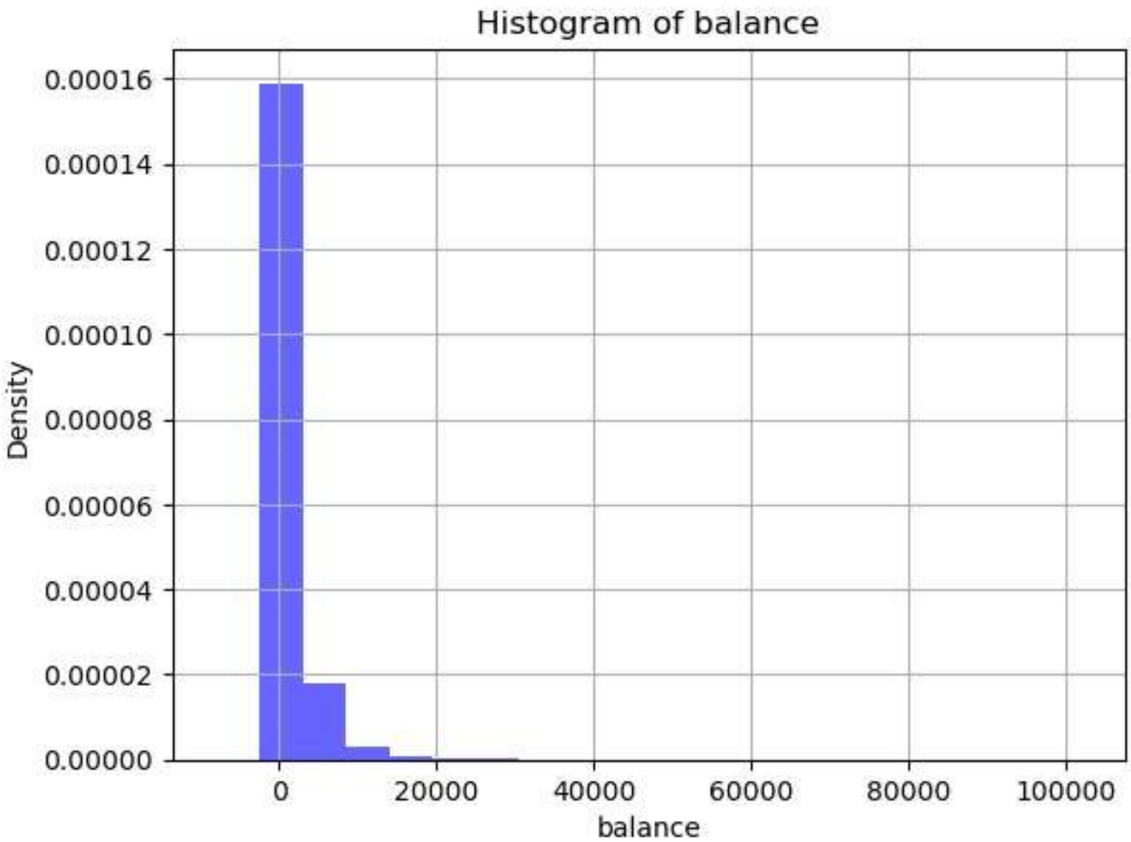

Boxplot of education
1
0.0
0.5
1.0
1.5
2.0

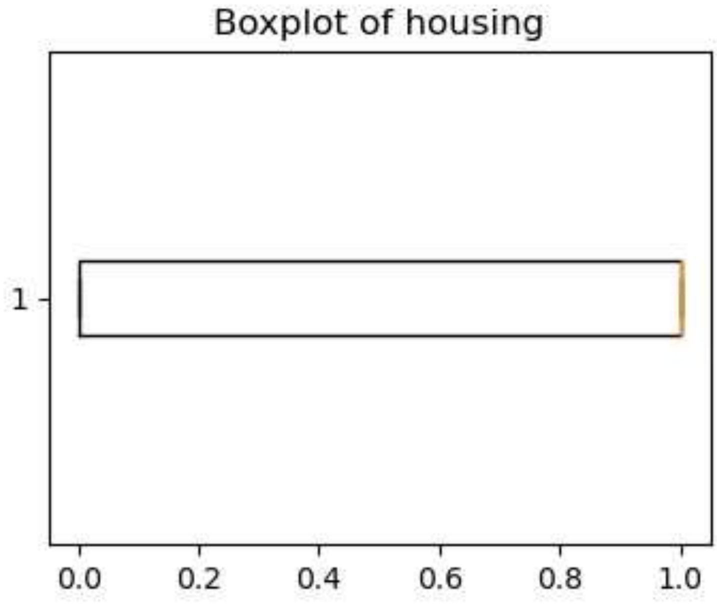

Histogram of education
Density
0
1
2
3
4
5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
education


Boxplot of default
1
0.0
0.2
0.4
0.6
0.8
1.0

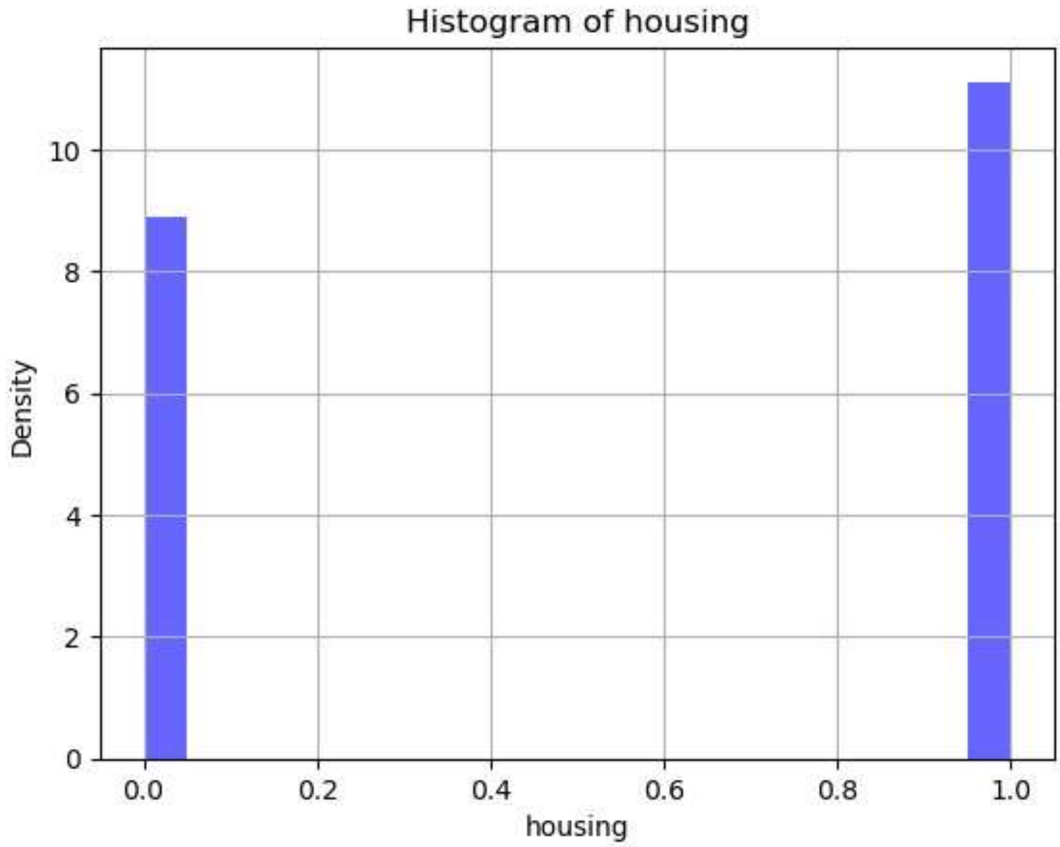

Histogram of default
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Density
0.0
0.2
0.4
0.6
0.8
1.0
default


Boxplot of balance
1
0
20000
40000
60000
80000
100000

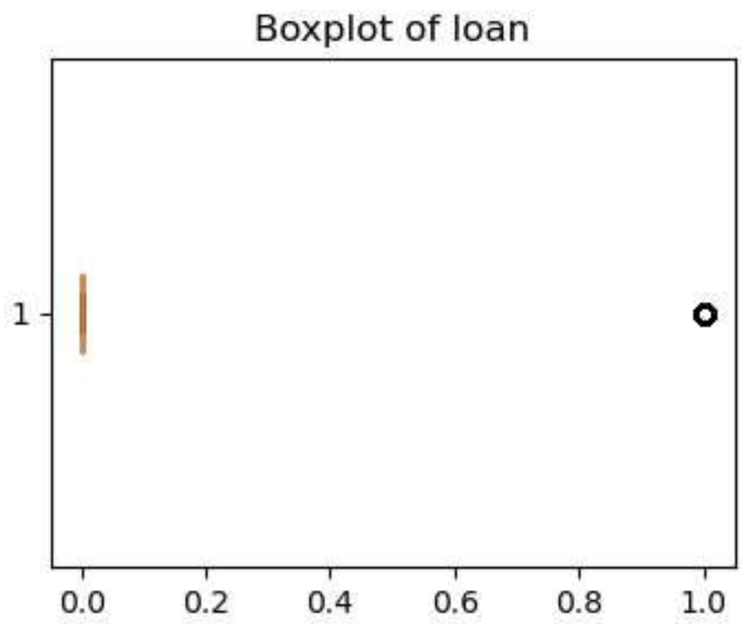

Histogram of balance
Density
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
20000
40000
60000
80000
100000
balance

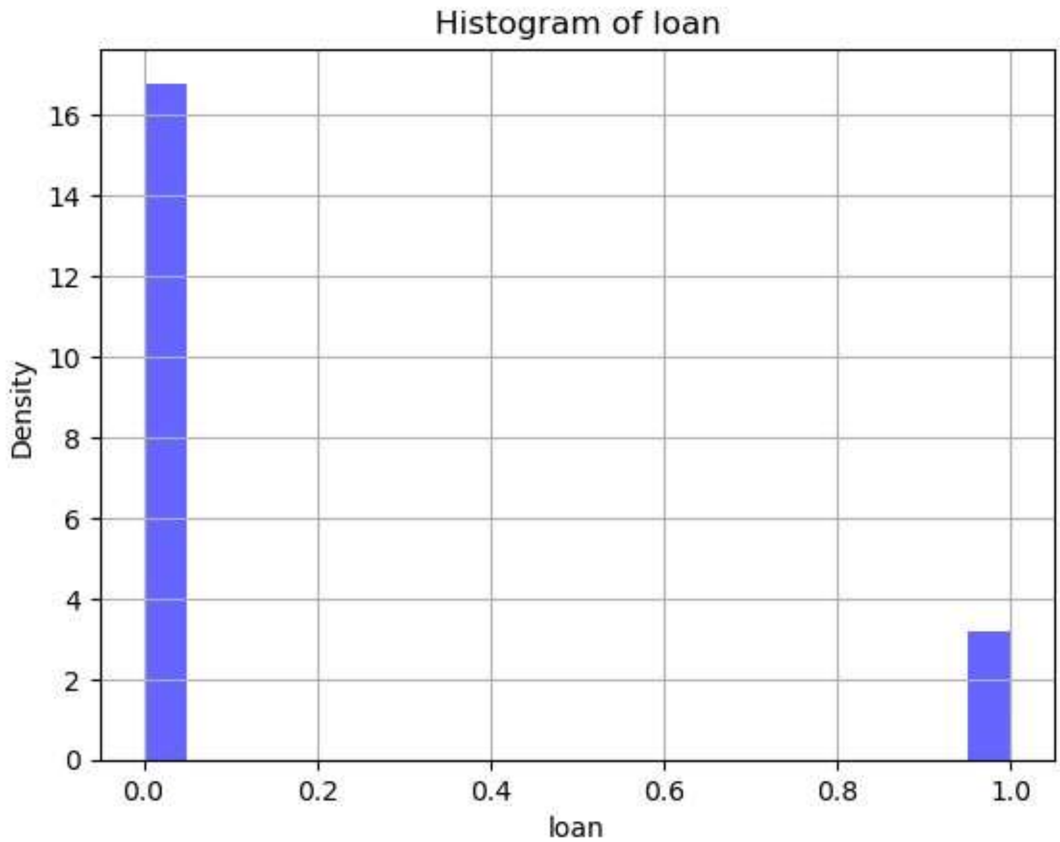


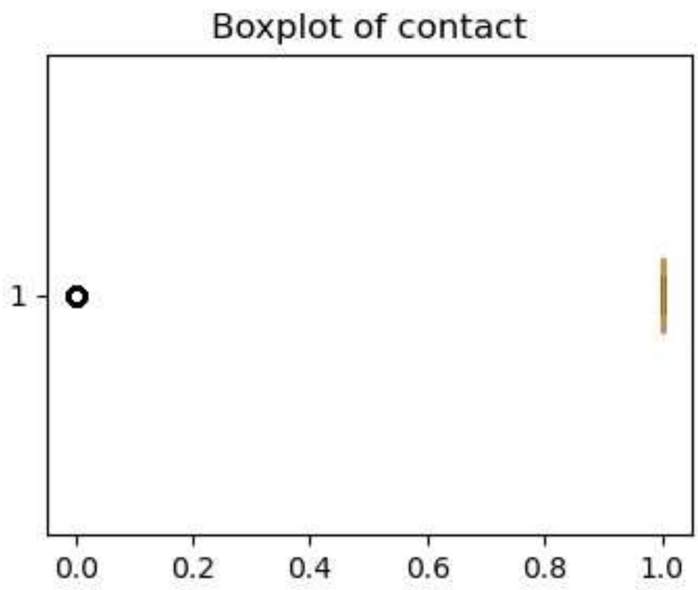
Boxplot of housing
1
0.0
0.2
0.4
0.6
0.8
1.0

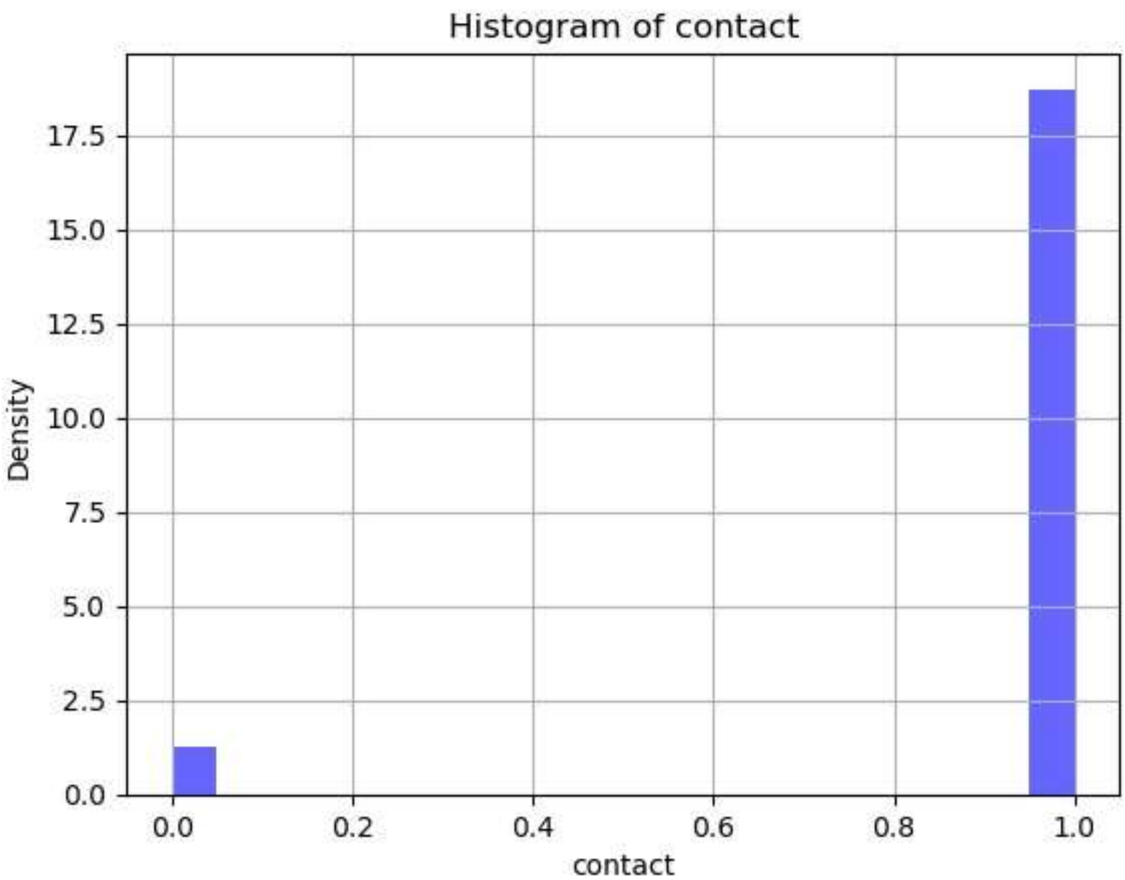


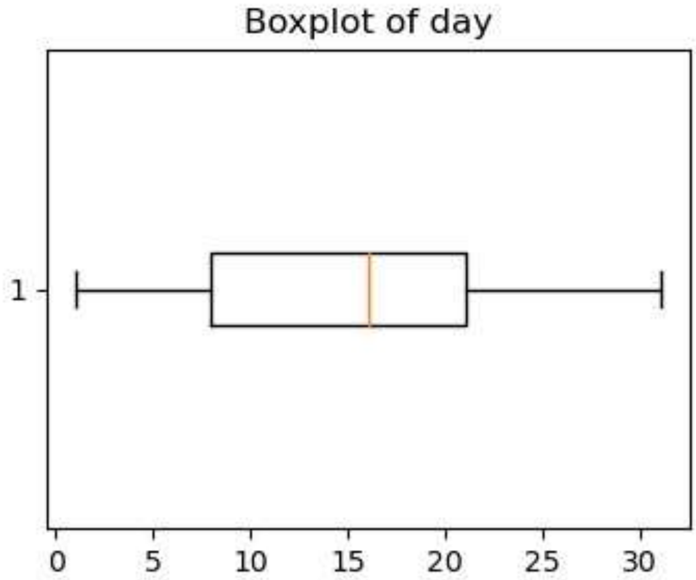
Histogram of housing
Density
10
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0
housing


Boxplot of loan
1
0.0
0.2
0.4
0.6
0.8
1.0

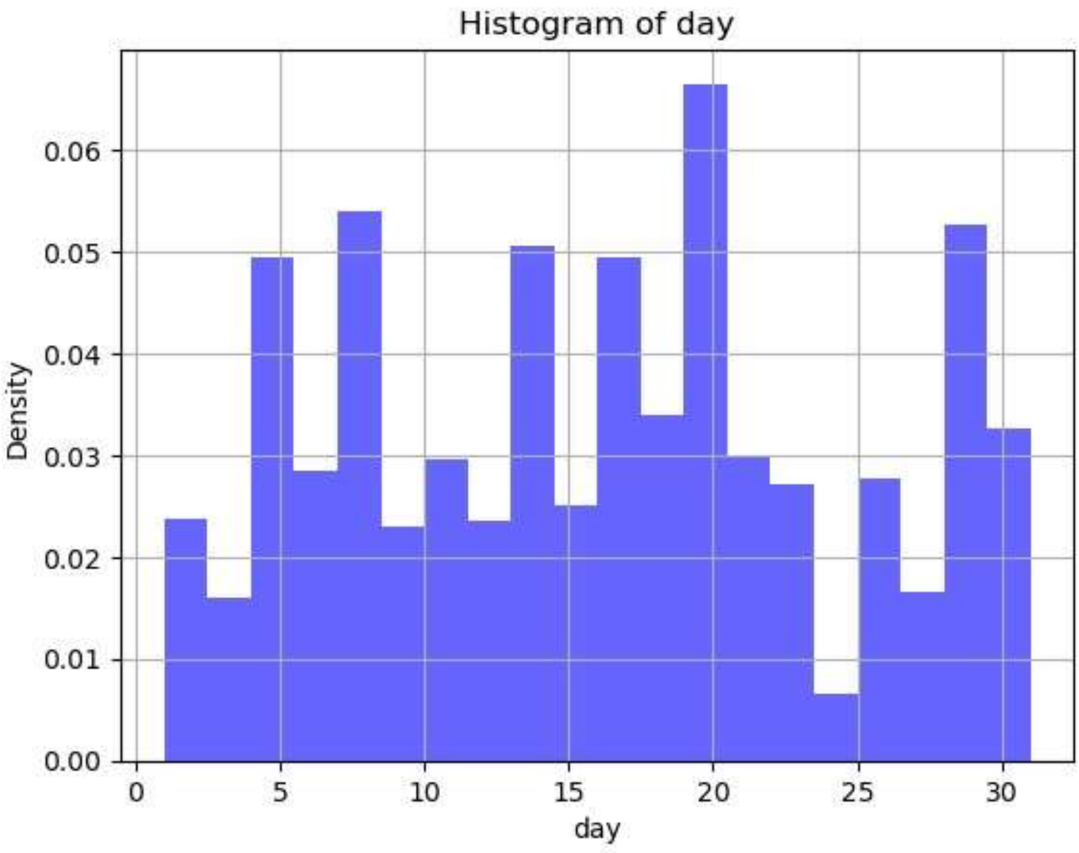

Histogram of loan
Density
0
2
4
6
8
10
12
14
16
0.0
0.2
0.4
0.6
0.8
1.0
loan


Boxplot of contact
1
0.0
0.2
0.4
0.6
0.8
1.0

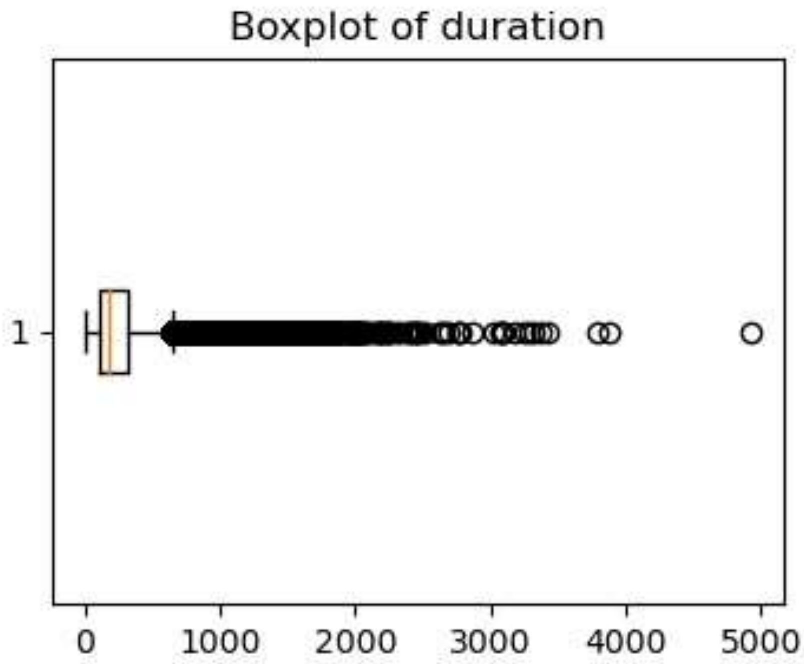

Histogram of contact
Density
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
contact

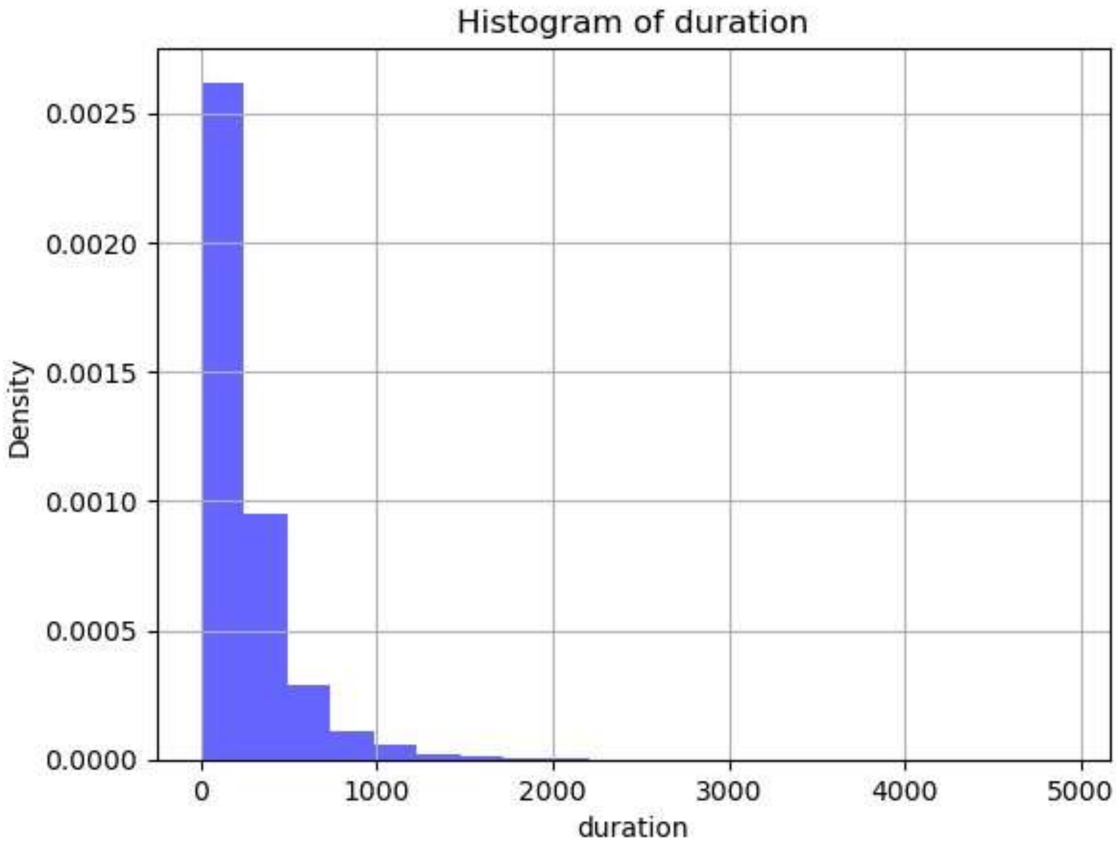


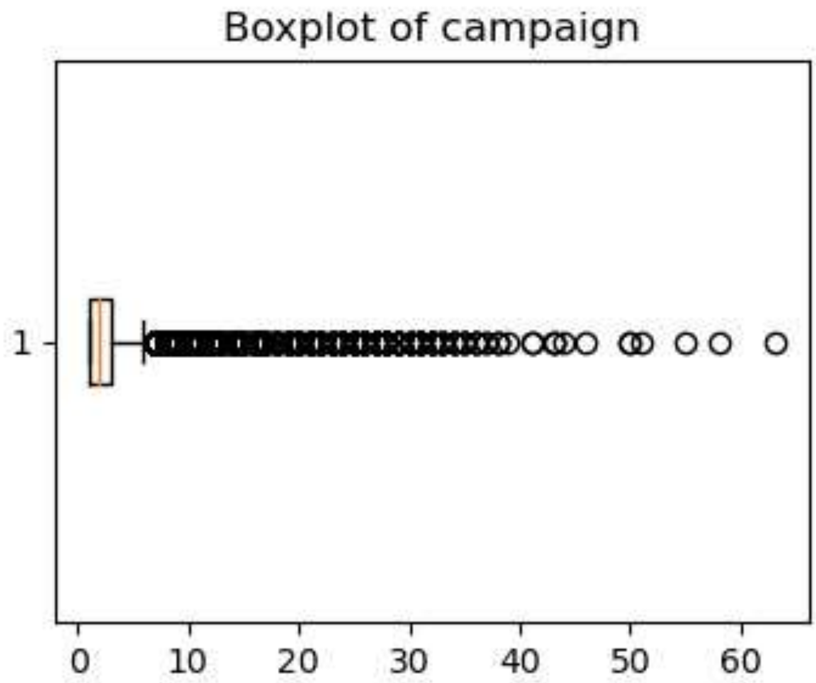
Boxplot of day
1
0
5
10
15
20
25
30

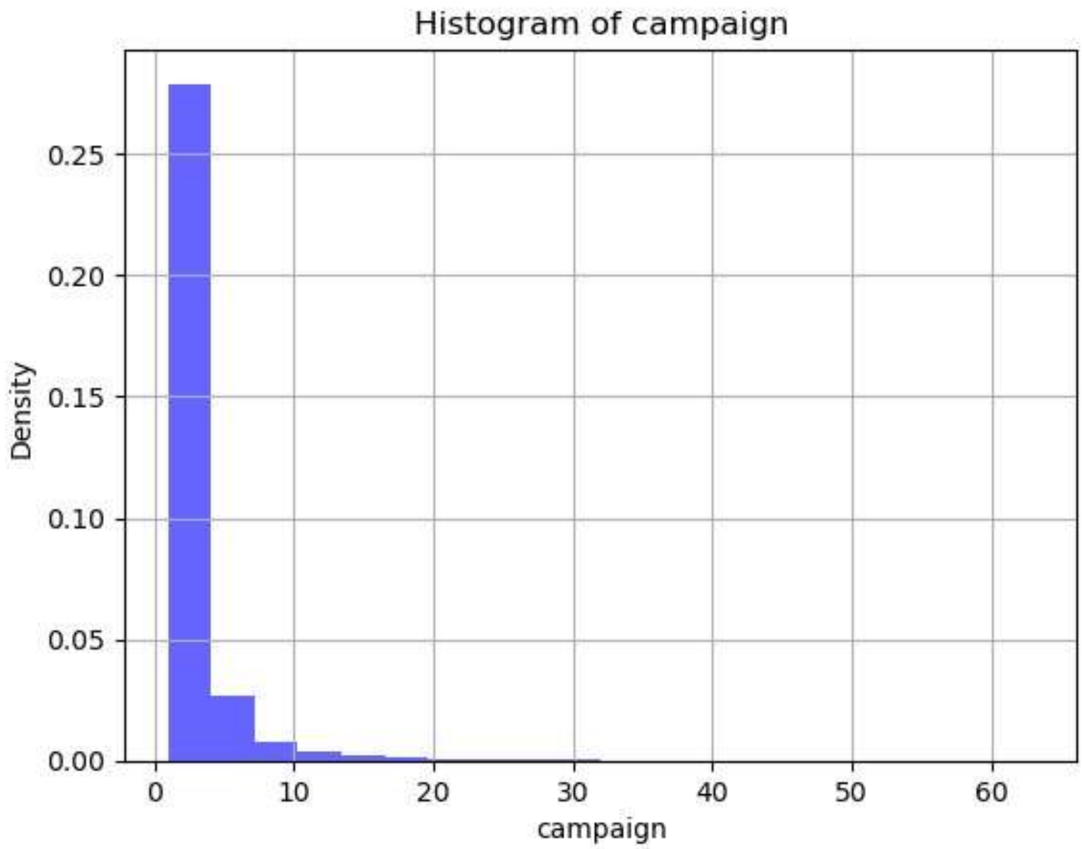


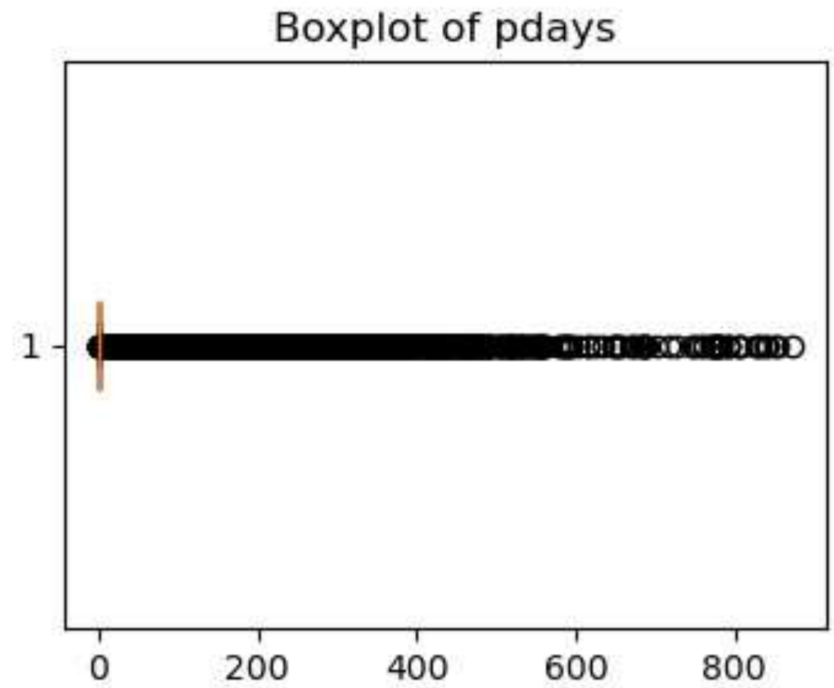
Histogram of day
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
5
10
15
20
25
30
day

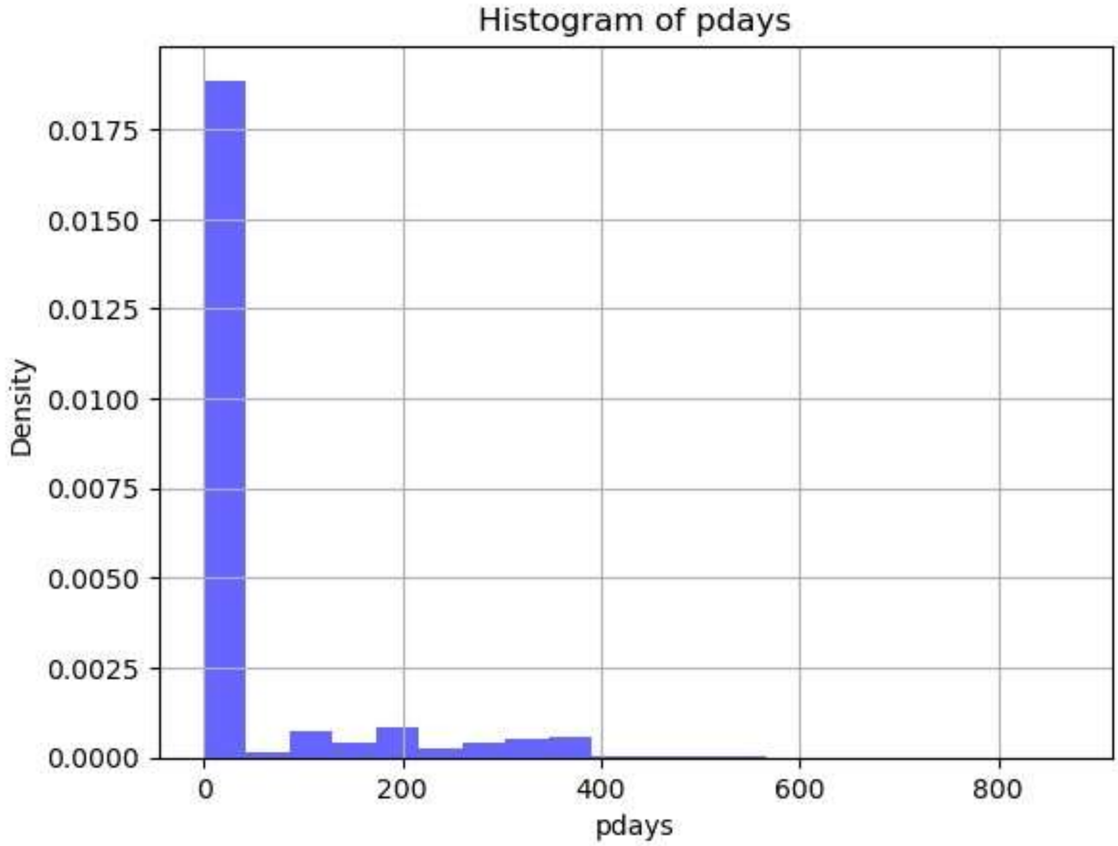

Boxplot of duration
1
0
1000
2000
3000
4000
5000

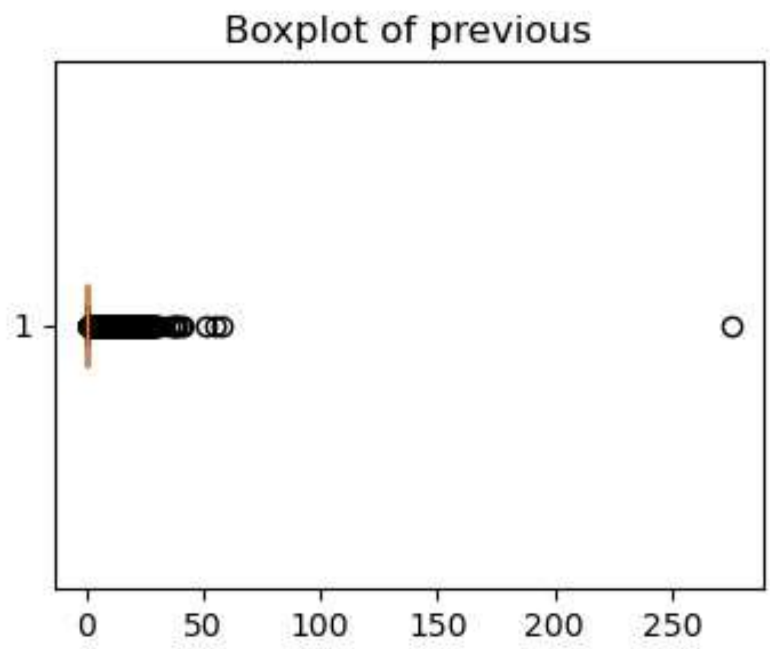

Histogram of duration
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
1000
2000
3000
4000
5000
duration

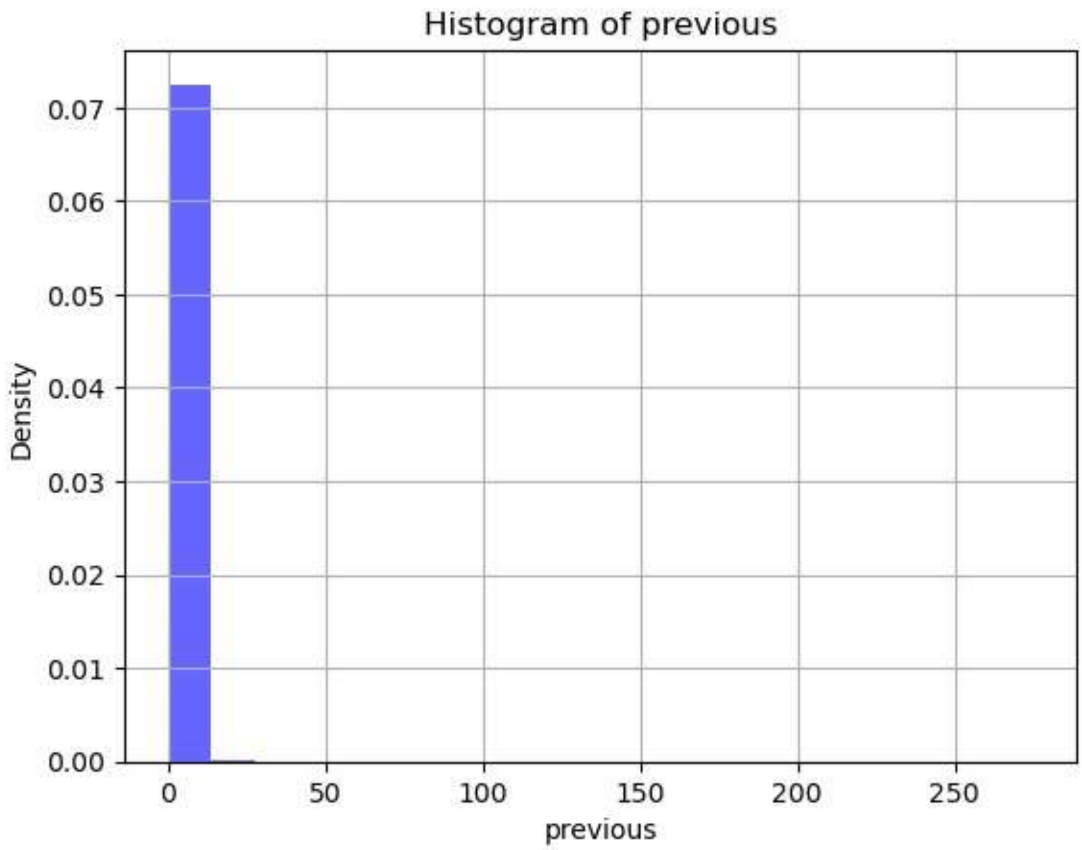

Boxplot of campaign
1
0
10
20
30
40
50
60

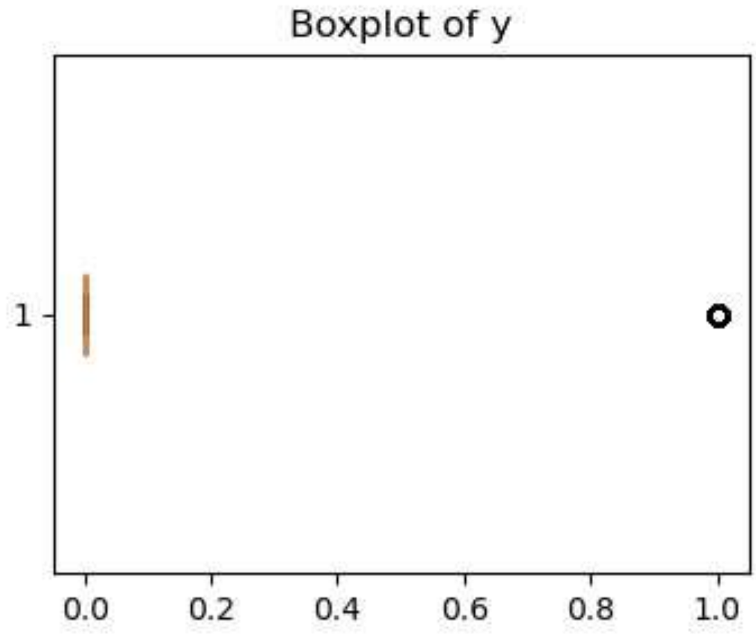

Histogram of campaign
Density
0.25
0.20
0.15
0.10
0.05
0.00
0
10
20
30
40
50
60
campaign

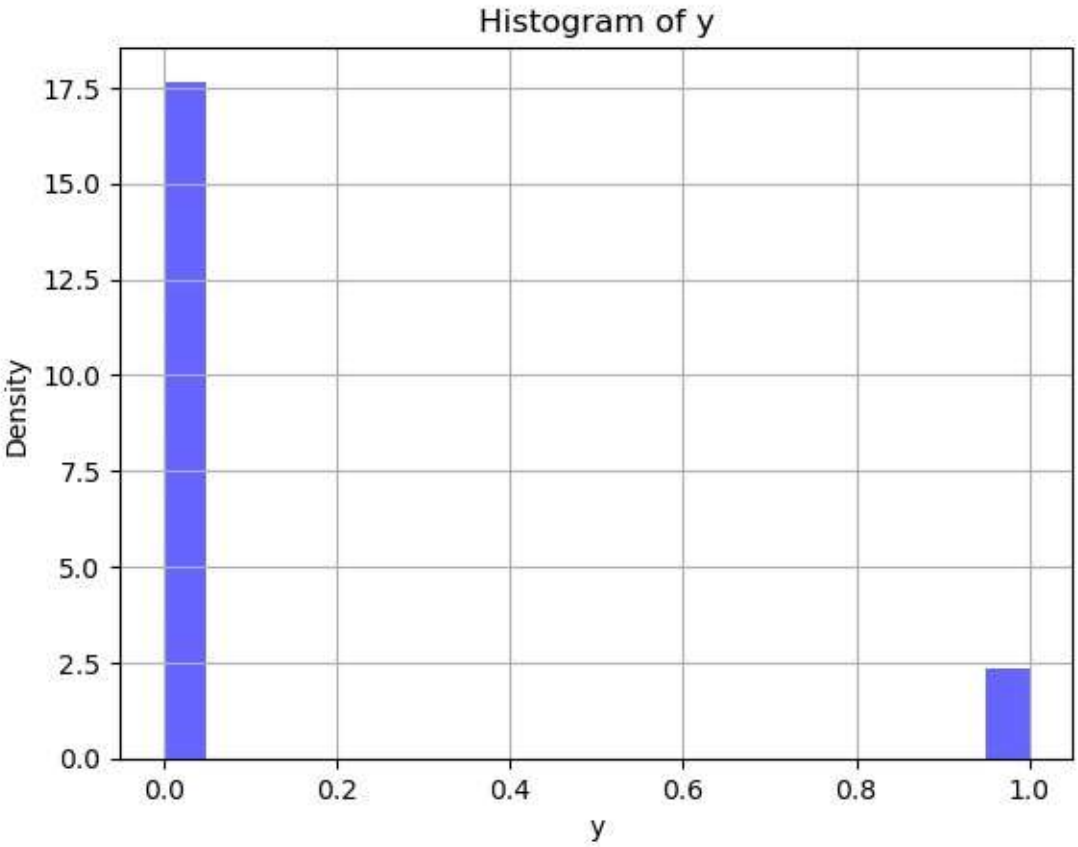

Boxplot of pdays
1
0
200
400
600
800


Histogram of pdays
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Density
0
200
400
600
800
pdays


Boxplot of previous
1
0
50
100
150
200
250


Histogram of previous
Density
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
50
100
150
200
250
previous


Boxplot of y
1
0.0
0.2
0.4
0.6
0.8
1.0


Histogram of y
Density
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
y

3. Machine Learning Models

3.1. KNN (K-Nearest Neighbours)

In the domain of predicting the effectiveness of bank marketing campaigns, the K-Nearest Neighbors (KNN) algorithm emerges as a promising methodology. Utilizing a feature space defined by diverse attributes like demographics, financial behaviors, and historical engagement patterns, KNN endeavors to classify whether a client is likely to subscribe to a term deposit or not. This process involves comparing the feature vectors of new clients with those of existing ones in the training dataset, identifying the K nearest neighbors based on a specified distance metric, and assigning the majority class label among these neighbors to the new client. Despite its simplicity, KNN can yield valuable insights into the dynamics of campaign effectiveness, particularly in scenarios characterized by nonlinear feature-churn relationships or where interpretability is crucial. However, it's imperative for practitioners to fine-tune parameters such as the value of K and carefully consider computational demands, particularly in scenarios involving large-scale datasets. Nevertheless, KNN remains a versatile tool in the arsenal of predictive analytics, providing an intuitive and interpretable approach to predicting the effectiveness of bank marketing campaigns.

3.2 Logistic Regression

Logistic Regression is a cornerstone of our project for forecasting the efficacy of bank marketing campaigns. In this context, it functions by estimating the probability of a client subscribing to a term deposit based on a multitude of predictor variables. These variables encompass demographic details, financial indicators, and temporal factors, collectively providing insight into client behavior and preferences.

One of the primary advantages of logistic regression in our project lies in its simplicity and interpretability. By modeling the relationship between features and campaign effectiveness, logistic regression offers a clear understanding of the factors influencing client decisions to subscribe to term deposits. This transparency is invaluable for banks aiming to tailor their marketing strategies effectively.

Furthermore, logistic regression's ability to handle nonlinear relationships between features and campaign outcomes makes it well-suited for our predictive task. By analyzing the coefficients associated with each feature, we can discern their relative importance in influencing client behavior. This insight aids in prioritizing marketing efforts and optimizing resource allocation for maximum impact.

3.3 SVM (Support Vector Machine)

In our project, Support Vector Machine (SVM) emerges as a potent tool for forecasting the effectiveness of bank marketing campaigns. SVM operates by delineating the optimal hyperplane that segregates the data into distinct classes, in this case, clients who subscribe to term deposits and those who do not. By maximizing the margin between these classes, SVM seeks to identify the most effective decision boundary based on various client features.

In the realm of campaign effectiveness prediction, SVM endeavors to discern the optimal decision boundary that segregates clients likely to subscribe to term deposits from those less inclined to do so. This is achieved by leveraging various client features such as demographics, financial behaviors, and historical engagement patterns. By mapping the data into a higher-dimensional space using kernel functions, SVM can effectively capture complex relationships between features and campaign outcomes, facilitating the identification of nonlinear decision boundaries.

The flexibility of SVM to capture intricate patterns in client data makes it particularly suitable for our predictive task. By accurately delineating the decision boundary, SVM aims to generalize well to unseen data, enabling precise predictions regarding client subscription behavior. However, it's essential to note that SVMs can be computationally intensive, especially with large datasets, and require meticulous parameter tuning for optimal performance. Nonetheless, with careful parameter optimization and feature engineering, SVMs hold the potential to offer valuable insights into campaign effectiveness, aiding financial institutions in optimizing their marketing strategies and maximizing subscription rates.

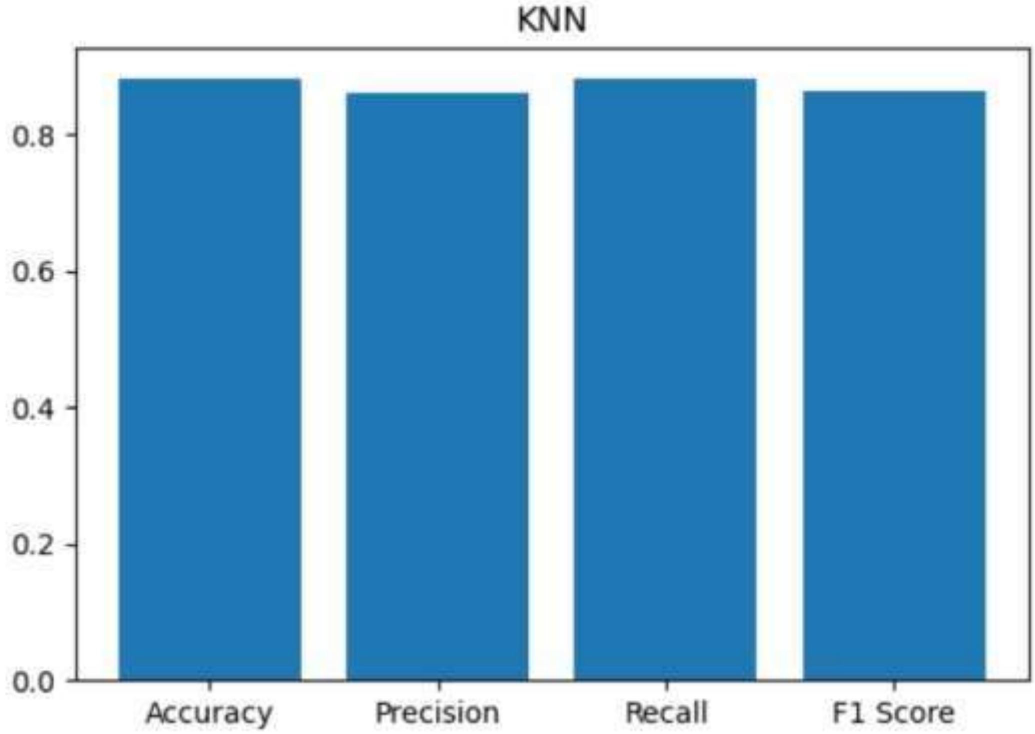
3.4 Kmeans

In the context of campaign effectiveness prediction, K-Means seeks to identify distinct groups of clients with similar characteristics and behaviors. By clustering clients based on features such as demographics, financial indicators, and past interaction patterns, K-Means provides valuable insights into heterogeneous client segments that may exhibit varying propensities to subscribe to term deposits.

The flexibility of K-Means to uncover hidden patterns in client data makes it particularly suitable for our predictive task. By grouping clients into clusters with similar subscription behaviors, K-Means facilitates the identification of key client segments that are more likely to respond positively to marketing campaigns. However, it's essential to note that K-Means clustering requires careful consideration of the number of clusters (K) and may not perform optimally in the presence of non-linear relationships between features.

Results:

4.1. KNN

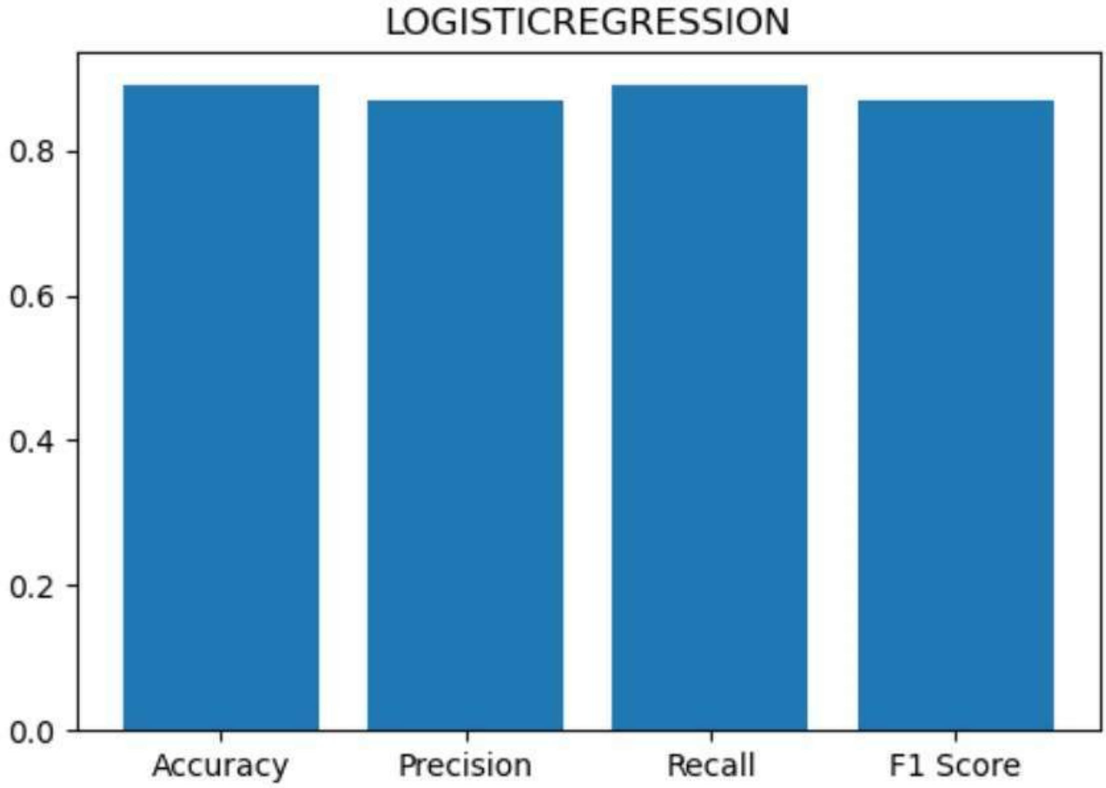


Confusion Matrix:

```
[[7668  276]
 [ 803  296]]
```

	precision	recall	f1-score	support
0	0.91	0.97	0.93	7944
1	0.52	0.27	0.35	1099
accuracy			0.88	9043
macro avg	0.71	0.62	0.64	9043
weighted avg	0.86	0.88	0.86	9043

4.2. Logistic Regression



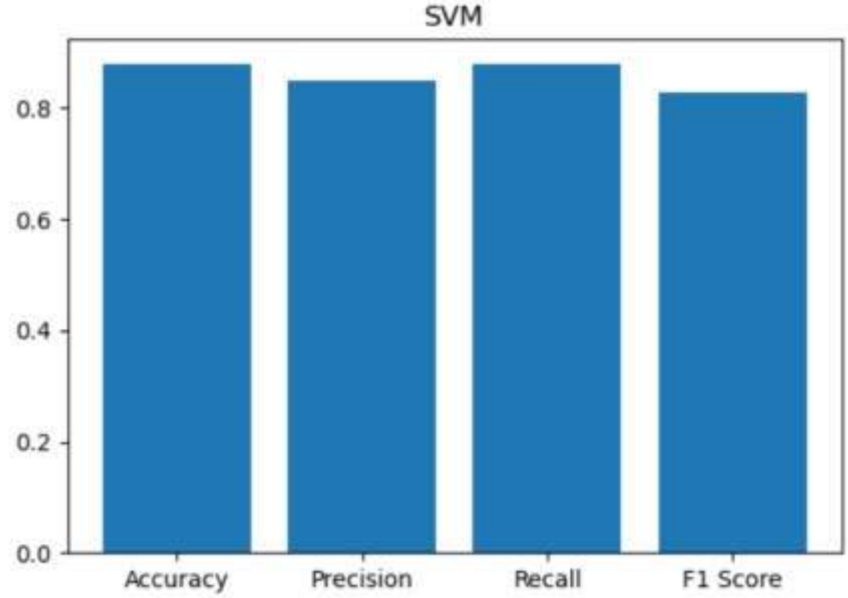
Confusion Matrix::

```
[[7834  110]
 [ 897  202]]
```

	precision	recall	f1-score	support
0	0.90	0.99	0.94	7944
1	0.65	0.18	0.29	1099
accuracy			0.89	9043
macro avg	0.77	0.58	0.61	9043
weighted avg	0.87	0.89	0.86	9043

Figure 3 Logistic Regression Result

4.3 SVM



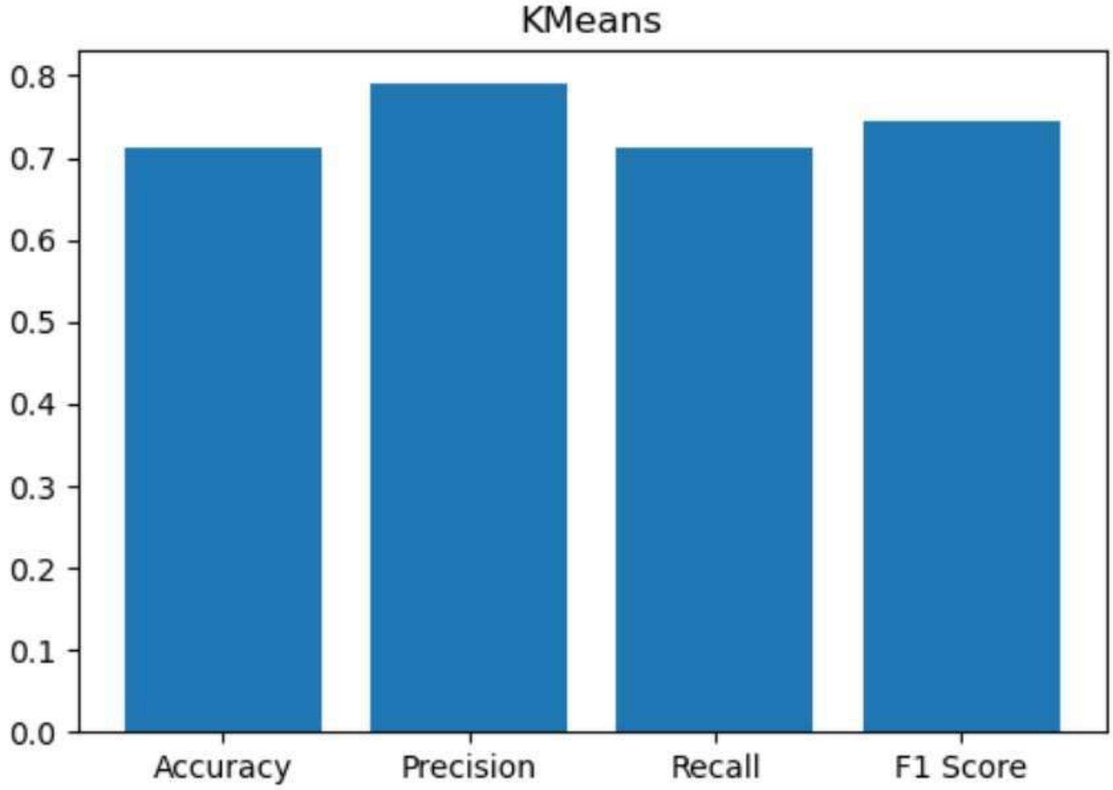
Confusion Matrix :

```
[[7941    3]
 [1097    2]]
```

	precision	recall	f1-score	support
0	0.88	1.00	0.94	7944
1	0.40	0.00	0.00	1099
accuracy			0.88	9043
macro avg	0.64	0.50	0.47	9043
weighted avg	0.82	0.88	0.82	9043

Figure 4 Support Vector Machine Result

4.4 K-Means:

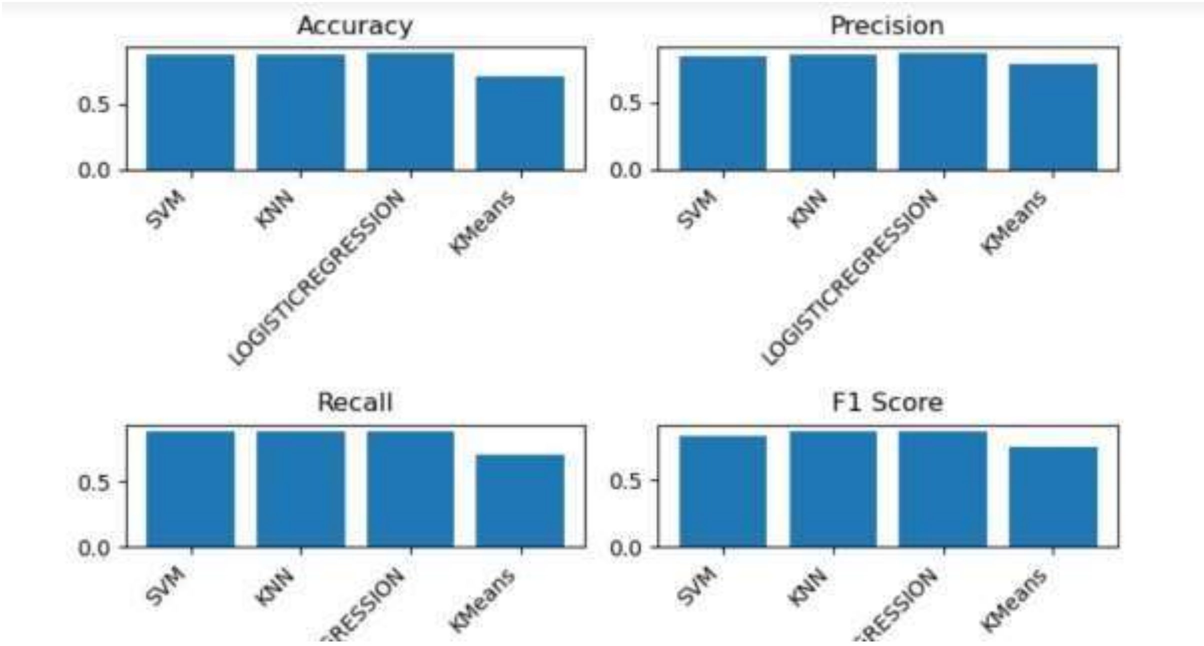


Confusion Matrix:

```
[[7944    0]
 [1099    0]]
```

Classification report:

	precision	recall	f1-score	support
0	0.88	1.00	0.94	7944
1	0.00	0.00	0.00	1099
accuracy			0.88	9043
macro avg	0.44	0.50	0.47	9043
weighted avg	0.77	0.88	0.82	9043


Accuracy
Precision
Recall
F1 Score
0.0
0.5
SVM
KNN
LOGISTICREGRESSION
KMeans

Concluding Remarks

Model	Accuracy	Precision	Recall	F1 Score
SVM	0.879575	0.850356	0.879575	0.827157
KNN	0.881455	0.859081	0.881455	0.864535
LOGISTICREGRESSION	0.890191	0.869367	0.890191	0.867404
KMeans	0.711158	0.790686	0.711158	0.745773

The analysis provides valuable insights into the factors influencing customer subscription to bank term deposits and highlights crucial trends in customer behavior. Here are some conclusion remarks based on the observations:

1. Demographic Patterns: Understanding the age distribution and occupation types of clients reveals nuances in subscription b The analysis provides valuable insights into the factors influencing customer subscription to bank term deposits and highlights crucial trends in customer behavior. Here are some conclusion remarks based on the observations:

1. Demographic Patterns: Understanding the age distribution and occupation types of clients reveals nuances in subscription behavior. While the average client age spans a wide range, the peak subscription age range suggests a specific demographic preference for term deposits, possibly influenced by financial stability and life stage considerations.

2. Marital Status and Education: Marital status and education levels play significant roles in subscription likelihood, indicating potential correlations between financial literacy, stability, and commitment levels.

3. Financial Obligations: The absence of defaults and housing loans positively influences subscription rates, while the presence of personal loans or multiple loan types deters subscription, reflecting the impact of financial constraints and risk aversion.

4. Communication Channels and Frequency: Contact through cellular phones appears to be more effective in prompting subscriptions, with a diminishing return observed beyond three contact attempts. This underscores the importance of targeted and timely communication strategies.

5. Seasonal Variations: Seasonality affects subscription rates, with May standing out as a particularly high-performing month. Understanding these seasonal fluctuations can inform marketing strategies and resource allocation.

6. Occupational and Educational Profiles: Clients with managerial roles and higher education levels exhibit higher subscription rates, suggesting a link between financial sophistication and propensity to invest in term deposits.

7. Engagement Metrics: Longer phone conversations correlate with higher subscription rates, indicating a potential relationship between engagement and interest in financial products.

8. Debt Status and Financial Freedom: Debt-free clients are more likely to subscribe, emphasizing the importance of financial stability and capacity for investment.

9. Model Performance and Recommendations: Machine learning models, particularly Logistic Regression and SVM, show promising accuracy rates in predicting churn. Leveraging ensemble methods and incorporating additional features could further enhance predictive power and robustness.

10. Business Implications: Businesses can use these insights to refine customer retention strategies, tailor communication channels, and optimize resource allocation. Proactively addressing churn risks and fostering stronger customer relationships are essential for long-term success in a competitive market landscape.

In conclusion, by leveraging the multifaceted insights gleaned from this analysis, businesses can make informed decisions to mitigate churn, enhance customer satisfaction, and drive sustainable growth. Continuous monitoring, adaptation, and innovation in predictive analytics remain critical for staying ahead in an ever-evolving market environment.ehavior. While the average client age spans a wide range, the peak subscription age range suggests a specific demographic preference for term deposits, possibly influenced by financial stability and life stage considerations.

2. Marital Status and Education: Marital status and education levels play significant roles in subscription likelihood, indicating potential correlations between financial literacy, stability, and commitment levels.

3. Financial Obligations: The absence of defaults and housing loans positively influences subscription rates, while the presence of personal loans or multiple loan types deters subscription, reflecting the impact of financial constraints and risk aversion.

4. Communication Channels and Frequency: Contact through cellular phones appears to be more effective in prompting subscriptions, with a

diminishing return observed beyond three contact attempts. This underscores the importance of targeted and timely communication strategies.

5. Seasonal Variations: Seasonality affects subscription rates, with May standing out as a particularly high-performing month. Understanding these seasonal fluctuations can inform marketing strategies and resource allocation.

6. Occupational and Educational Profiles: Clients with managerial roles and higher education levels exhibit higher subscription rates, suggesting a link between financial sophistication and propensity to invest in term deposits.

7. Engagement Metrics: Longer phone conversations correlate with higher subscription rates, indicating a potential relationship between engagement and interest in financial products.

8. Debt Status and Financial Freedom: Debt-free clients are more likely to subscribe, emphasizing the importance of financial stability and capacity for investment.

9. Model Performance and Recommendations: Machine learning models, particularly Logistic Regression and SVM, show promising accuracy rates in predicting churn. Leveraging ensemble methods and incorporating additional features could further enhance predictive power and robustness.

10. Business Implications: Businesses can use these insights to refine customer retention strategies, tailor communication channels, and optimize resource allocation. Proactively addressing churn risks and fostering stronger customer relationships are essential for long-term success in a competitive market landscape.

In conclusion, by leveraging the multifaceted insights gleaned from this analysis, businesses can make informed decisions to mitigate churn, enhance customer satisfaction, and drive sustainable growth. Continuous monitoring, adaptation, and innovation in predictive analytics remain critical for staying ahead in an ever-evolving market environment.

References :

1.Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, Burlington (2000).

2. Pujari, A.K.: Data Mining Techniques, 1st edn. Universities Press (India) Private Limited, Hyderabad (2001)

ml-project

May 9, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
df=pd.read_csv('bank-full.csv',sep=";")
df
```

[2]:
```
       age          job  marital  education default  balance housing loan  \
0      58   management married    tertiary   no    2143  yes   no
1      44   technician single secondary no    29    yes   no 2  33
       entrepreneur    married secondary     no    2     yes yes 3
            47    blue-collar      married    unknown    no    1506
            yes   no
 4      33      unknown   single   unknown     no       1     no   no
 …    …          …       …       …      …       …      … …
45206  51   technician married    tertiary   no    825   no    no
45207  71   retired divorced primary     no    1729  no    no
45208  72   retired married secondary  no    5715  no    no
45209  57   blue-collar      married secondary     no    668   no    no
45210  37 entrepreneur married secondary     no    2971  no    no
        contact day month duration campaign pdays previous poutcome y

 0        unknown   5   may      261         1     -1        0 unknown    no

 1        unknown   5   may      151         1     -1        0 unknown    no

 2        unknown   5   may       76         1     -1        0 unknown    no

 3        unknown   5   may       92         1     -1        0 unknown    no

 4        unknown   5   may      198         1     -1        0 unknown    no

 …            … …    …        …        …     …        …       … …

 45206   cellular  17   nov      977         3     -1         0 unknown yes

 45207   cellular  17   nov      456         2     -1         0 unknown yes

 45208   cellular  17   nov     1127         5    184         3 success yes

 45209 telephone  17   nov      508         4     -1        0 unknown    no

 45210    cellular 17   nov      361         2    188        11    other   no

[45211 rows x 17 columns]
```

[3]:
```
df.columns
```

[3]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
         'loan', 'contact', 'day', 'month', 'duration', 'campaign',
         'pdays',
         'previous', 'poutcome', 'y'],
        dtype='object')
```

[4]:
```
df.info()
```

```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to
45210 Data columns (total 17
columns):
#   Column     Non-Null Count
               Dtype
--- ------     -------------- ----
               -
0   age        45211 non-null
               int64
1   job        45211      non-null
               object
2   marital    45211      non-null
               object
 3  education  45211 non-null object
 4  default    45211      non-null
               object
 5  balance    45211 non-null
               int64
 6  housing    45211      non-null
               object
 7  loan       45211      non-null
               object
 8  contact    45211      non-null
               object
 9  day        45211 non-null
               int64
 10 month      45211      non-null
               object
 11 duration   45211 non-null
               int64
 12 campaign   45211 non-null
               int64
 13 pdays      45211 non-null
               int64
```

```
 14 previous   45211 non-null
               int64
 15 poutcome   45211      non-null
               object
 16 y          45211      non-null
               object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

[5]:
```
df.head()
```

[5]:
```
   age       job marital education default balance housing loan \
0     58    management married    tertiary    no    2143  yes
 no 1  44    technician single secondary no    29    yes    no
2   33 entrepreneur married secondary   no    2     yes yes
3   47 blue-collar married    unknown     no    1506  yes    no
4   33 unknown single   unknown     no    1     no    no

  contact day month duration campaign pdays previous poutcome y
 0 unknown    5   may      261         1     -1         0 unknown no
 1 unknown    5   may      151         1     -1         0 unknown no
 2 unknown    5   may       76         1     -1         0 unknown no
 3 unknown    5   may       92         1     -1         0 unknown no
 4 unknown    5   may      198         1     -1         0 unknown no
```

[6]:
```
df.tail()
```

[6]:
```
        age         job marital education default balance housing loan \
45206   51    technician married     tertiary    no    825    no    no
45207   71    retired divorced primary     no    1729  no    no
45208   72    retired married secondary   no    5715  no    no
45209   57    blue-collar      married secondary      no    668    no    no
45210   37 entrepreneur married secondary     no    2971  no    no

       contact day month duration campaign pdays previous poutcome  y
 45206  cellular  17   nov      977         3     -1         0 unknown yes
 45207  cellular  17   nov      456         2     -1         0 unknown yes
 45208  cellular  17   nov     1127         5    184         3 success yes
 45209 telephone 17   nov      508         4     -1        0 unknown    no
 45210  cellular  17   nov      361         2    188        11    other   no
```

[7]:
```
df.shape
```

[7]:
```
(45211, 17)
```

[8]:
```
df.describe()
```

[8]:
```
age    balance     day   duration    campaign \ count 45211.000000
        45211.000000 45211.000000 45211.000000 45211.000000
 mean     40.936210   1362.272058    15.806419   258.163080    2.763841
```

```
std      10.618762   3044.765829     8.322476   257.527812   3.098021
min      18.000000 -8019.000000     1.000000     0.000000   1.000000
25%      33.000000     72.000000     8.000000   103.000000   1.000000
50%      39.000000    448.000000    16.000000   180.000000   2.000000
75%      48.000000   1428.000000    21.000000   319.000000   3.000000
max      95.000000 102127.000000    31.000000  4918.000000  63.000000

               pdays     previous
count 45211.000000 45211.000000
mean  40.197828  0.580323 std
      100.128746 2.303441 min
      -1.000000  0.000000 25%
      -1.000000  0.000000
50% -1.000000 0.000000 75% -
1.000000     0.000000     max
871.000000 275.000000
```

[9]:
```
df.describe(include='all')
```

[9]:
```
                  age         job marital education default      balance \
count   45211.000000      45211   45211     45211   45211  45211.000000
unique           NaN         12       3         4       2           NaN
top              NaN blue-collar married secondary      no           NaN
freq             NaN       9732   27214     23202   44396           NaN
mean       40.936210        NaN     NaN       NaN     NaN   1362.272058
std        10.618762        NaN     NaN       NaN     NaN   3044.765829
min        18.000000        NaN     NaN       NaN     NaN -8019.000000
25%        33.000000        NaN     NaN       NaN     NaN     72.000000
50%        39.000000        NaN     NaN       NaN     NaN    448.000000
75%        48.000000        NaN     NaN       NaN     NaN   1428.000000
max         95.000000  NaN  NaN  NaN  NaN  102127.000000  housing  loan
      contact day month duration \

count   45211 45211   45211 45211.000000 45211 45211.000000
unique      2     2       3          NaN    12          NaN
top       yes    no cellular          NaN   may          NaN
freq  25130 37967     29285          NaN 13766          NaN
mean      NaN   NaN       NaN    15.806419   NaN   258.163080
std       NaN   NaN       NaN     8.322476   NaN   257.527812
min       NaN   NaN       NaN     1.000000   NaN     0.000000
25%       NaN   NaN       NaN     8.000000   NaN   103.000000
50%       NaN   NaN       NaN    16.000000   NaN   180.000000
75%       NaN   NaN       NaN    21.000000   NaN   319.000000
max       NaN  NaN  NaN  31.000000  NaN  4918.000000  campaign  pdays
         previous poutcome y

count 45211.000000 45211.000000 45211.000000   45211 45211
unique          NaN          NaN          NaN       4     2
```

```
top           NaN          NaN          NaN unknown     no
freq          NaN          NaN          NaN   36959  39922
mean     2.763841    40.197828     0.580323     NaN    NaN
std      3.098021   100.128746     2.303441     NaN    NaN
min      1.000000    -1.000000     0.000000     NaN    NaN
25%      1.000000    -1.000000     0.000000     NaN    NaN
50%      2.000000    -1.000000     0.000000     NaN    NaN
75%      3.000000    -1.000000     0.000000     NaN    NaN
max     63.000000   871.000000   275.000000     NaN    NaN
```

[10]:
```
count_duplicated = df.duplicated().sum()
print('Dataset having',count_duplicated,'duplicated
values')
```

```
Dataset having 0 duplicated values
```

[11]:
```
cat_var=[]
for var in df.columns:
    if df[var].dtype=='object':
        cat_var.append(var)
categorical_variables=np.array(cat_var)
categorical_variables
```

[11]:
```
array(['job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'poutcome', 'y'], dtype='<U9')
```

[12]:
```
num_var=[]
for var in df.columns:
    if df[var].dtype=='int64':
        num_var.append(var)
numerical_variables=np.array(num_var)
numerical_variables
```

[12]:
```
array(['age', 'balance', 'day', 'duration', 'campaign', 'pdays',
       'previous'], dtype='<U8')
```

[13]:
```
for var in df.columns:
    print(df[var].value_counts())
```

```
32    2085
31    1996
33    1972
34    1930
35    1894
      …
93       2
90       2
95       2
88       2
94       1
Name: age, Length: 77, dtype: int64
```

```
blue-collar   9732
management    9458
technician    7597
admin.        5171
services      4154
retired       2264
self-employed 1579
entrepreneur  1487
unemployed    1303
housemaid     1240
student        938
unknown        288
Name: job, dtype:
int64 married
 27214 single
 12790 divorced
 5207
Name: marital, dtype:
int64 secondary  23202
tertiary   13301
primary     6851
unknown     1857
Name: education, dtype: int64
no 44396 yes
815
Name: default, dtype: int64
 0        3514
 1        195
 2        156
 4        139
 3        134
         …
-381       1
 4617      1
 20584     1
 4358      1
 16353     1
Name: balance, Length: 7168, dtype:
int64 yes   25130 no   20081
Name: housing, dtype: int64
no    37967
yes   7244
Name: loan, dtype:
int64 cellular
 29285 unknown
 13020 telephone  2906
```

```
Name: contact, dtype: int64
20   2752
18   2308
21   2026
17   1939
6    1932
5    1910
14   1848
8    1842
28   1830
7    1817
19   1757
29   1745
15   1703
12   1603
13   1585
30   1566
9    1561
11   1479
4    1445
16   1415
2    1293
27   1121
3    1079
26   1035
23    939
22    905
25    840
31    643
10    524
24    447
1     322
Name: day, dtype:
int64 may   13766
jul   6895 aug
 6247 jun   5341 nov
 3970 apr   2932 feb
 2649 jan   1403 oct
 738 sep    579 mar
 477 dec    214
Name: month, dtype: int64
124    188
90     184
89    177
104 175
122    175
```

```
       …
1833     1
1545     1
1352     1
1342     1
1556     1
Name: duration, Length: 1573, dtype: int64
1    17544
2    12505 3     5521
4       3522
5       1764
6       1291
7       735
8       540
9       327
10      266
11      201
12      155
13      133
14      93
15      84
16      79
17      69
18      51
19      44
20      43
21      35
22      23
25      22
23      22
24      20
29      16
28      16
26      13
31      12
27      10
32       9
30       8
33       6
34       5
36       4
35       4
43       3
38       3
37       2
50       2
```

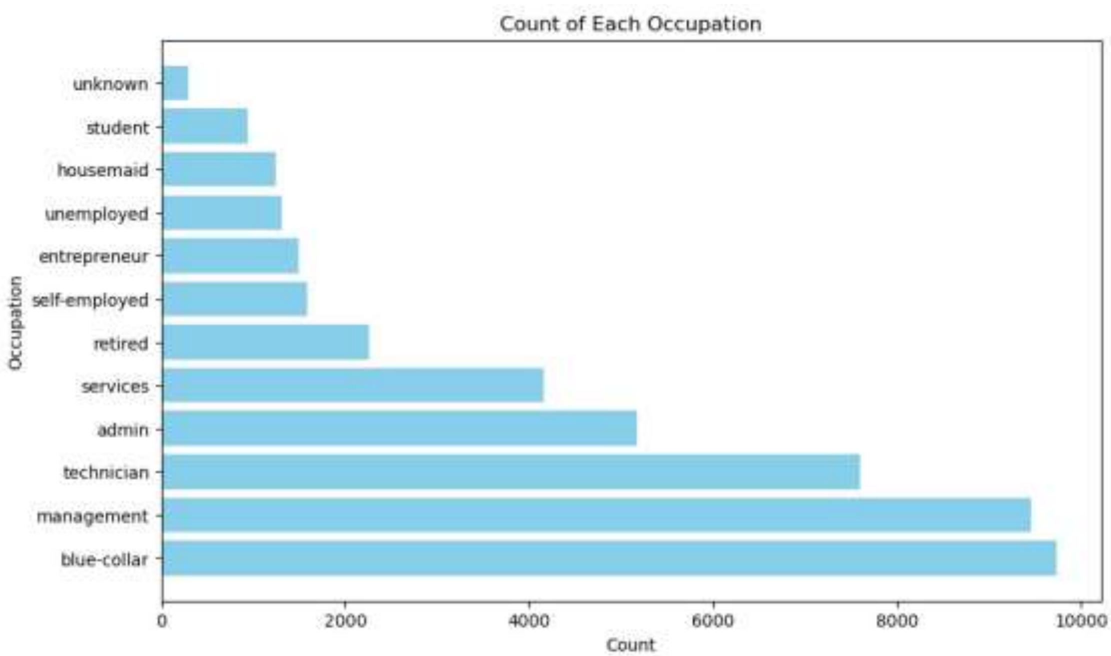
```
41       2
46       1
58       1
55       1
63       1
51       1
39       1
44       1
Name: campaign, dtype: int64
-1    36954
 182  167
 92     147
 91     126
 183    126
      …
 449      1
 452      1
 648      1
 595      1
 530      1
Name: pdays, Length: 559, dtype: int64
0        36954
1        2772
2        2106
3        1142
4        714
5        459
6        277
7        205
8        129
9        92
10       67
11       65
12       44
13       38
15       20
14       19
17       15
16       13
19        11
20        8
23        8
18        6
22        6
24        5
27        5
```

```
21        4
29        4
25        4
30        3
38        2
37        2
26        2
28        2
51        1
275       1
58        1
32        1
40        1
55        1
35        1
41        1
Name: previous, dtype: int64
unknown  36959
failure   4901
other     1840
success   1511
Name: poutcome, dtype: int64
no    39922
yes   5289
```

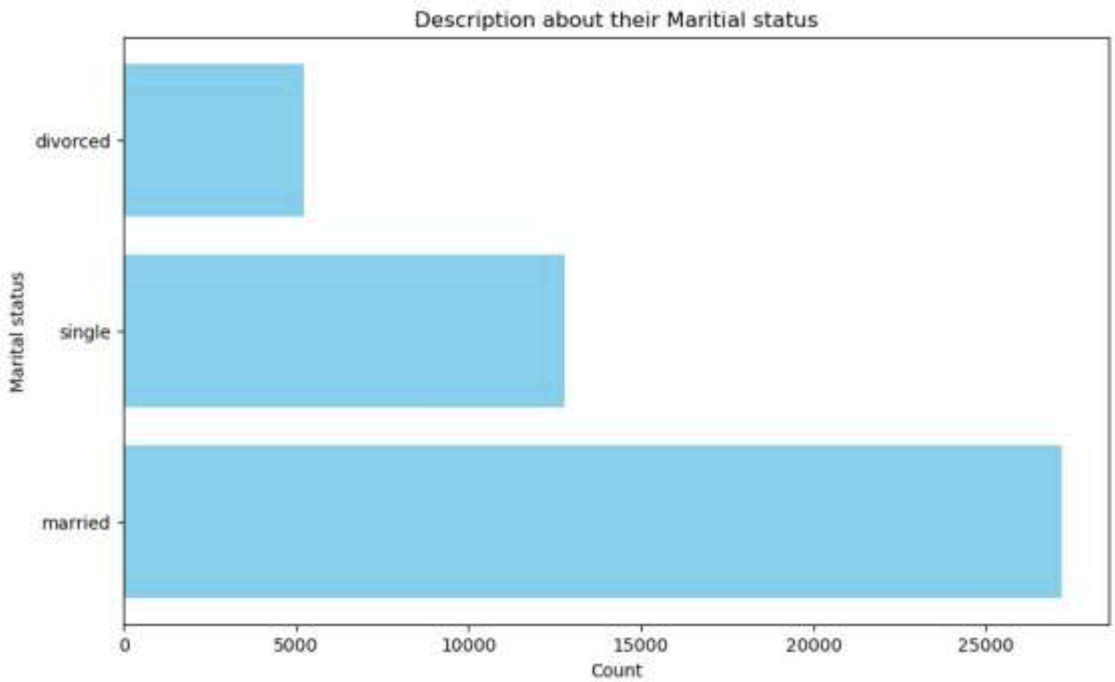
```
Name: y, dtype: int64
```

[14]:

```
data={
    'blue-collar':9732,
    'management':9458,
    'technician':7597,
    'admin':5171,
    'services':4154,
    'retired':2264,
    'self-employed':1579,
    'entrepreneur':1487,
    'unemployed':1303,
    'housemaid':1240,
    'student':938,
    'unknown':288
}
x_labels=list(data.keys())
y_labels=list(data.values())
plt.figure(figsize=(10, 6))
plt.barh(x_labels, y_labels, color='skyblue')
plt.xlabel('Count')
plt.ylabel('Occupation')
plt.title('Count of Each Occupation ')
plt.show()
```

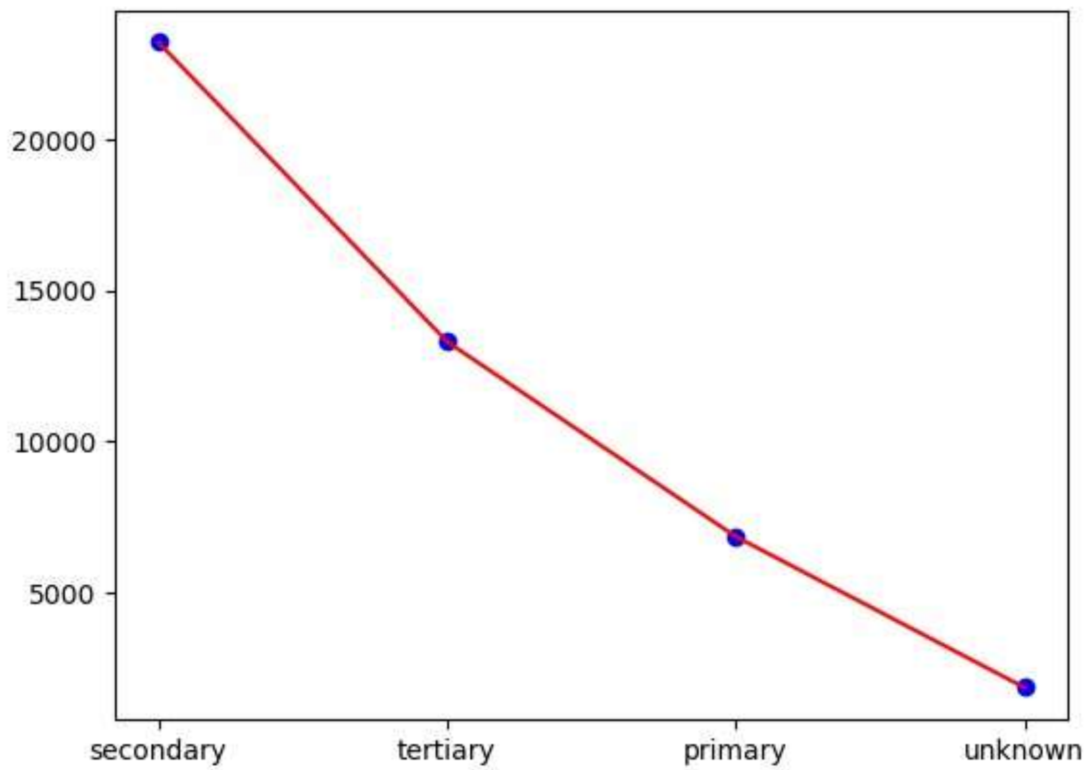


[15]:
```
dict={
    'married':27214,
    'single':12790,
    'divorced':5207
}
label_x=list(dict.keys())
label_y=list(dict.values())
plt.figure(figsize=(10, 6))
plt.barh(label_x,label_y, color='skyblue')
plt.xlabel('Count')
plt.ylabel('Marital status')
plt.title('Description about their Maritial status ')
plt.show()
```

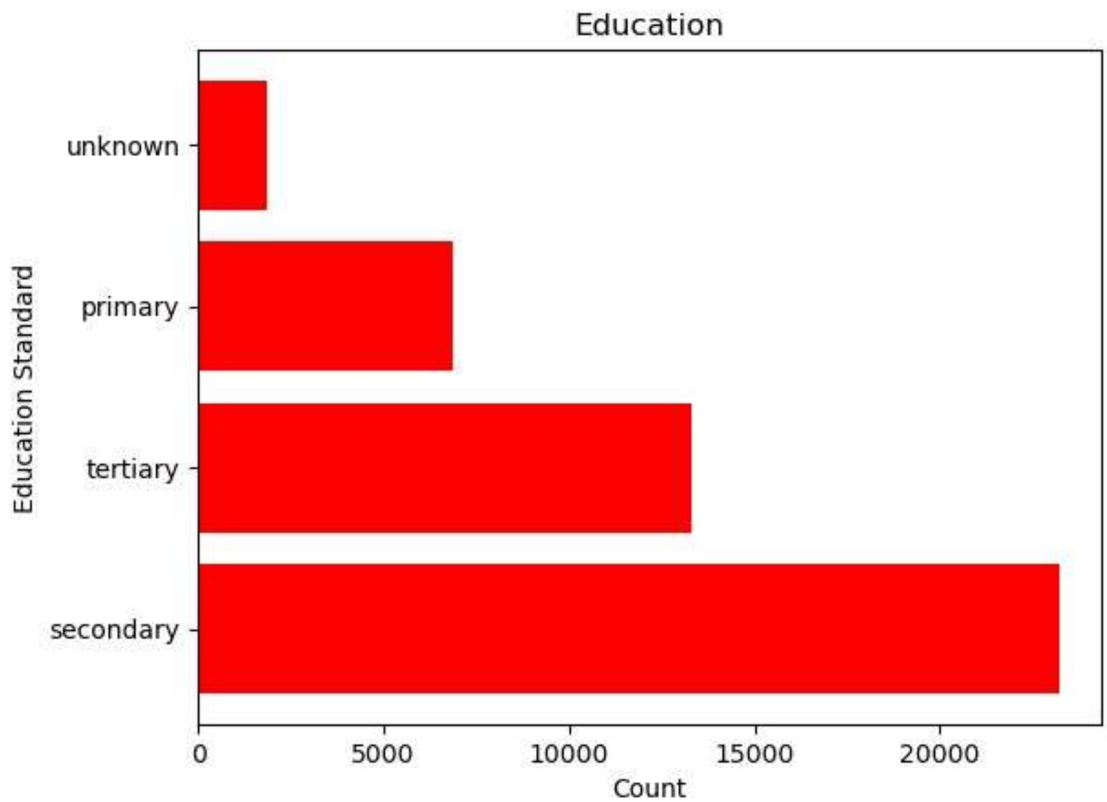


[16]:
```
d={
'secondary':23202,
'tertiary':13301,
'primary':6851,
'unknown':1857
}
u=list(d.keys())
v=list(d.values())
plt.scatter(u,v,color='blue')
```

```python
plt.plot(u,v,color='red')
plt.show()
```



[17]:
```python
plt.barh(u,v,color='red')
plt.xlabel('Count')
plt.ylabel('Education Standard')
plt.title('Education')
plt.show()
```



[18]:
```
df=df.replace('unknown',np.nan)
```

[19]:
```
da=pd.DataFrame({'columns':df.columns,'number_of_nulls_values':df.isna().
 ↪sum(),'percentage_null_values':round(df.isna().sum()*100/len(df),2)})
```

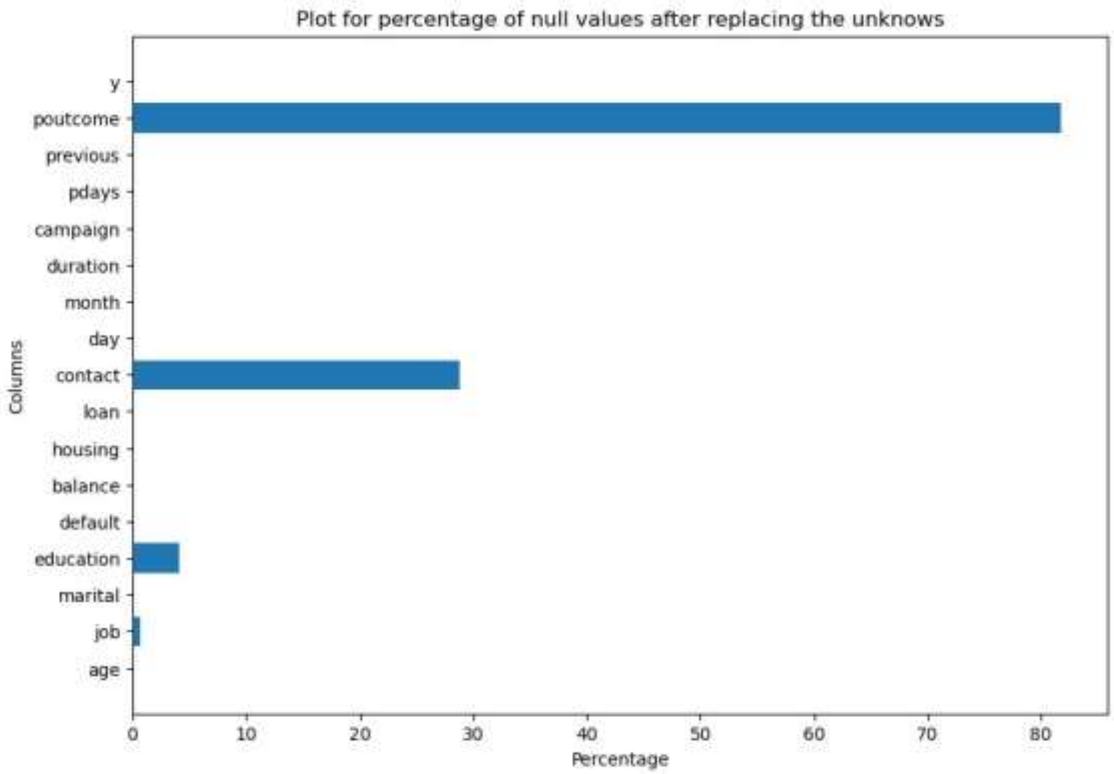
[20]:
```
da
```

[20]:

	columns	number_of_nulls_values	percentage_null_values
age	age	0	0.00
job	job	288	0.64
marital	marital	0	0.00
education	education	1857	4.11
default	default	0	0.00
balance	balance	0	0.00
housing	housing	0	0.00
loan	loan	0	0.00
contact	contact	13020	28.80
day	day	0	0.00
month	month	0	0.00
duration	duration	0	0.00
campaign	campaign	0	0.00

pdays	pdays	0	0.00
previous	previous	0	0.00
poutcome	poutcome	36959	81.75
y	y	0	0.00

[21]:
```
plt.figure(figsize=(10,7))
plt.barh('columns','percentage_null_values',data=da)
plt.xlabel('Percentage')
plt.ylabel('Columns')
plt.title('Plot for percentage of null values after replacing the unknows ')
plt.show()
```



[22]:
```
null_variables=['poutcome','contact','education','
job'] for x in null_variables:
    print(df[x].value_counts())
    
↪print('----------------------------------------------------------------------
-----'
```

```
failure  4901
other    1840
success  1511
Name: poutcome, dtype: int64
----------------------------------------------------------------------
-----------
---
cellular    29285
telephone   2906
Name: contact, dtype: int64
----------------------------------------------------------------------
-----------
---
secondary   23202
tertiary    13301
primary     6851
Name: education, dtype: int64
----------------------------------------------------------------------
-----------
---
blue-collar    9732
management     9458
technician     7597
admin.         5171
services       4154
retired        2264
self-employed  1579
entrepreneur   1487
unemployed     1303
housemaid      1240
student         938
Name: job, dtype: int64
----------------------------------------------------------------------
-----------
---
```

[23]:
```
df.drop(columns='poutcome',inplace=True)
df.shape
```

[23]:
```
(45211, 16)
```

[24]:
```
 df['contact']=df['contact'].fillna(df['contact'].mode()[0])
df['education']=df['education'].fillna(df['education'].mode()[0])
df['job']=df['job'].fillna(df['job'].mode()[0])
```

[25]:
```
 df.isna().sum()
```

[25]:
```
age        0
```

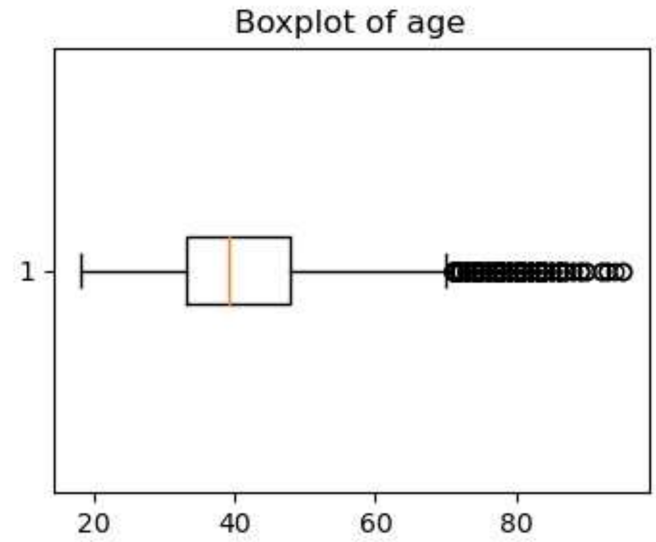
```
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
y            0
dtype: int64
```

[26]:
```
columns_mean=df.mean()
columns_mean
```

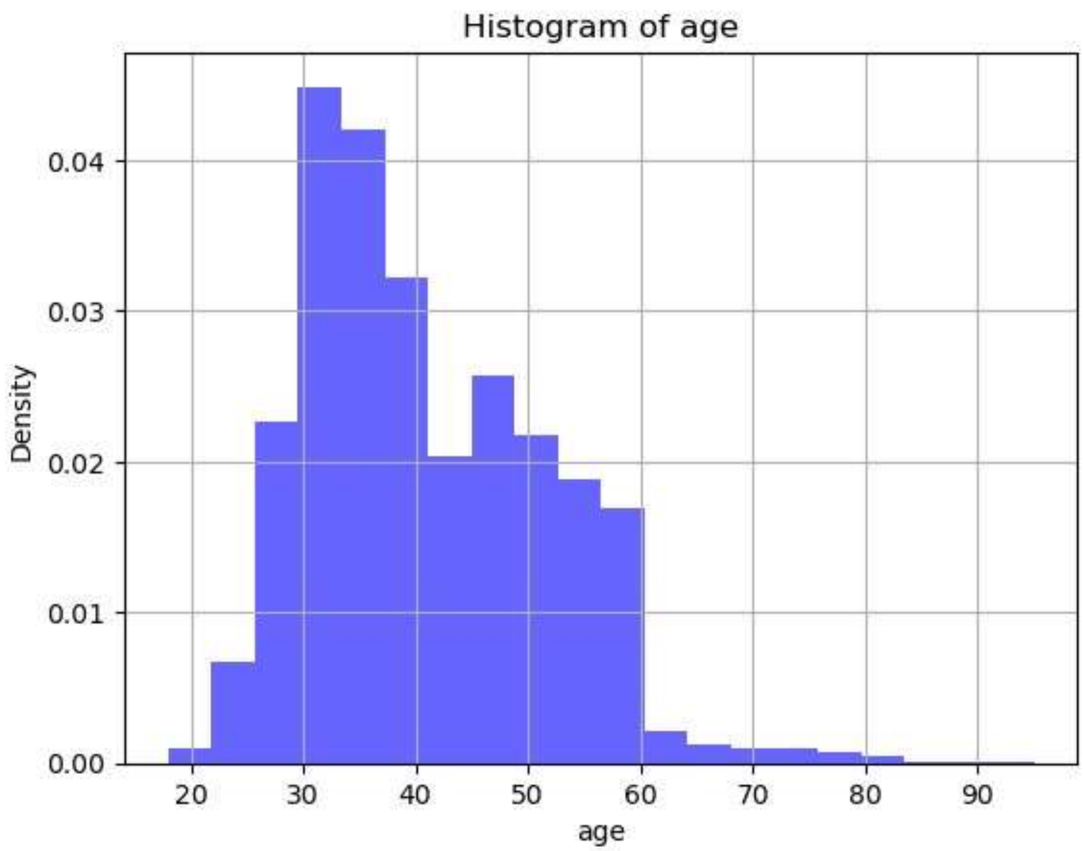
```
C:\Users\PHALGUN\AppData\Local\Temp\ipykernel_26316\4192548252.py:1:
FutureWarning: The default value of numeric_only in DataFrame.mean is
deprecated. In a future version, it will default to False. In
addition, specifying 'numeric_only=None' is deprecated. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  columns_mean=df.mean()
```

[26]:
```
age           40.936210
balance     1362.272058
day           15.806419
duration     258.163080
campaign       2.763841
pdays         40.197828
previous       0.580323
dtype: float64
```

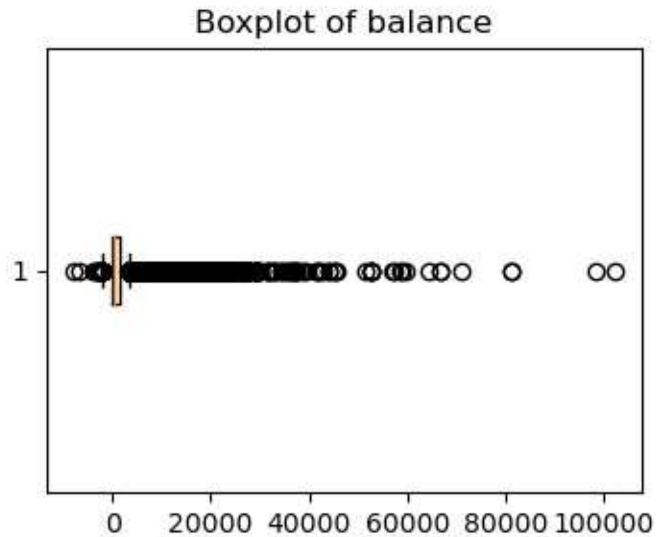
[27]:
```
for column in df.select_dtypes(include='int64'):
    plt.figure(figsize=(4,3))
    plt.boxplot(df[column], vert=False)
    plt.title(f'Boxplot of {column}')
    plt.show()

    plt.hist(df[column], bins=20, density=True, alpha=0.6, color='b')
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Density')
    plt.grid(True)
    plt.show()
```

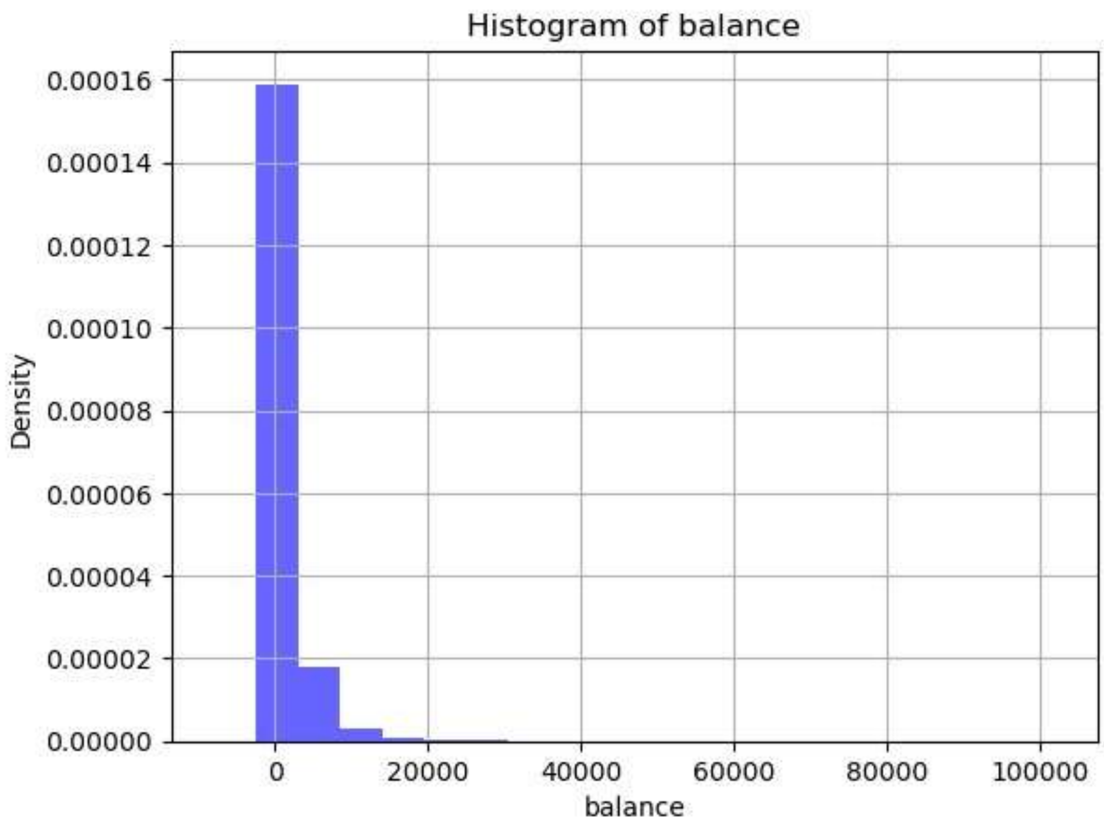


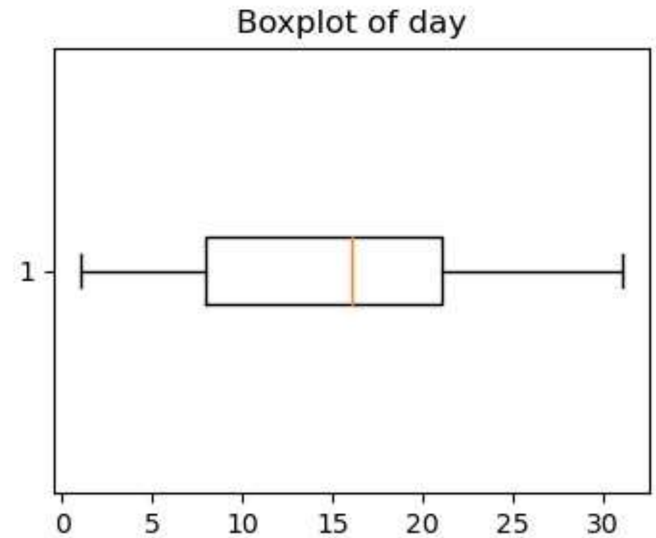
Boxplot of age
1
20
40
60
80



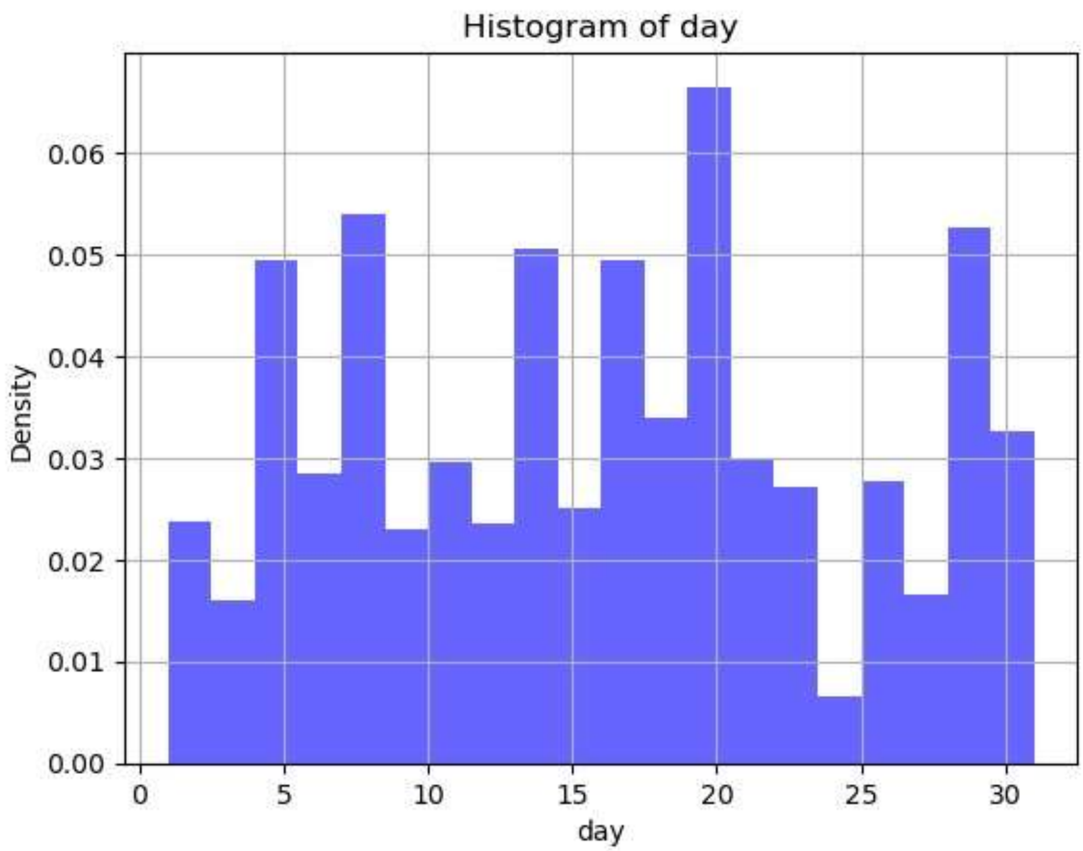
Histogram of age
Density
0.00
0.01
0.02
0.03
0.04
20
30
40
50
60
70
80
90
age


Boxplot of balance
1
0
20000
40000
60000
80000
100000

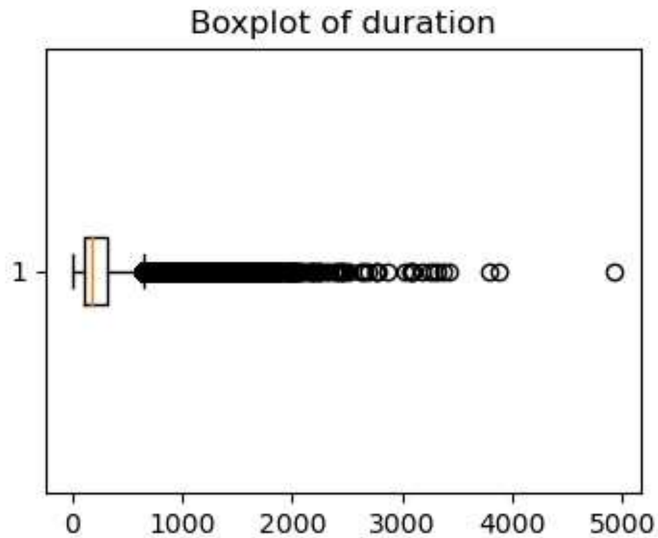

Histogram of balance
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
Density
0
20000
40000
60000
80000
100000
balance

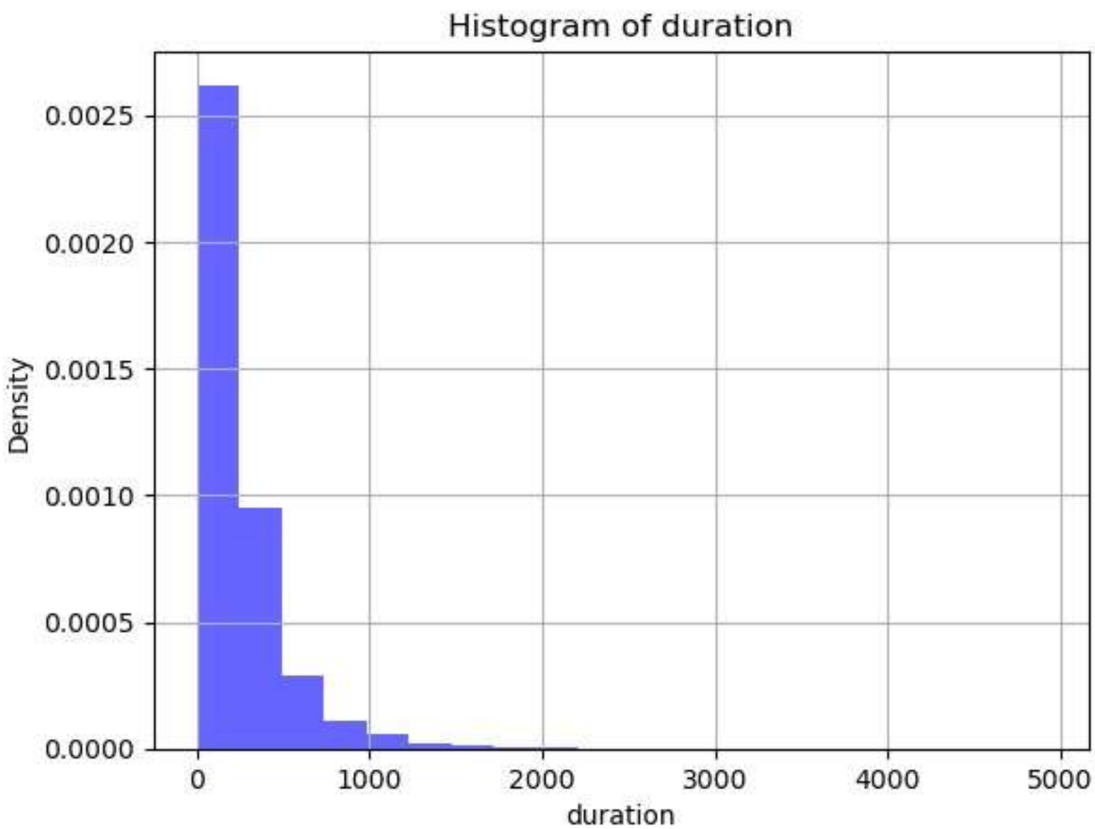


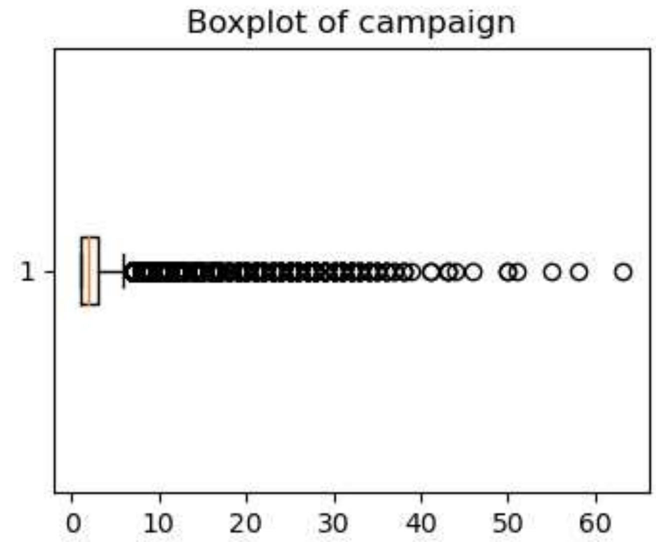
Boxplot of day
1
0
5
10
15
20
25
30

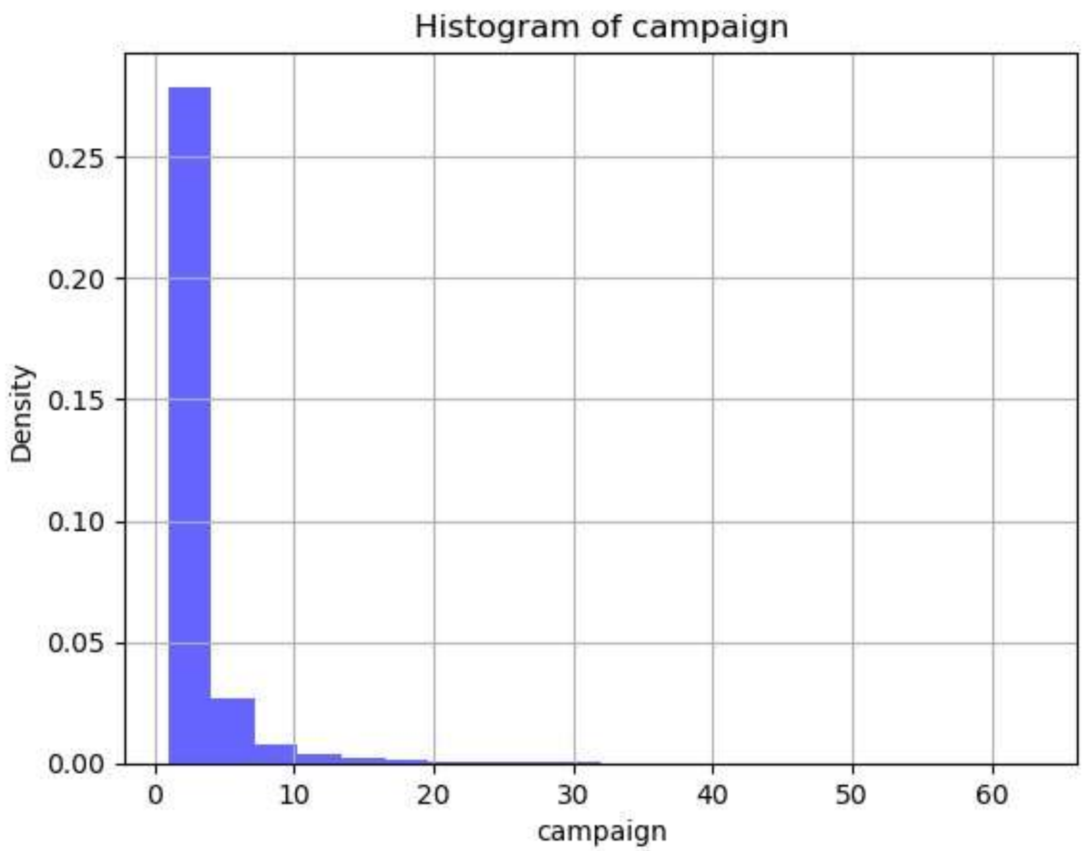


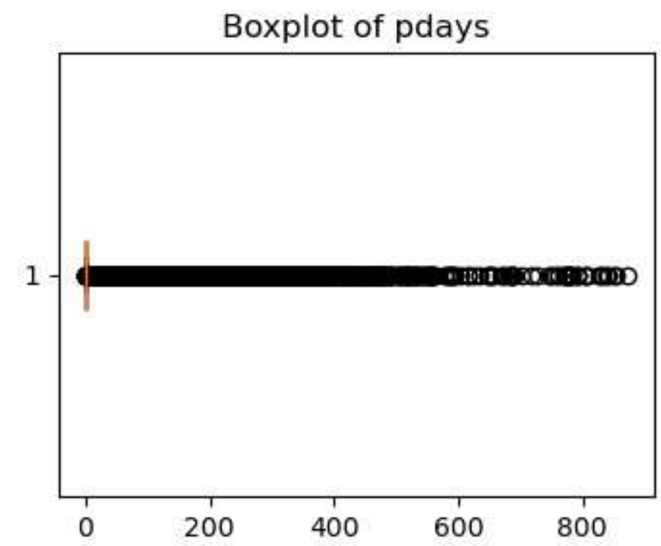
Histogram of day
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0
5
10
15
20
25
30
day

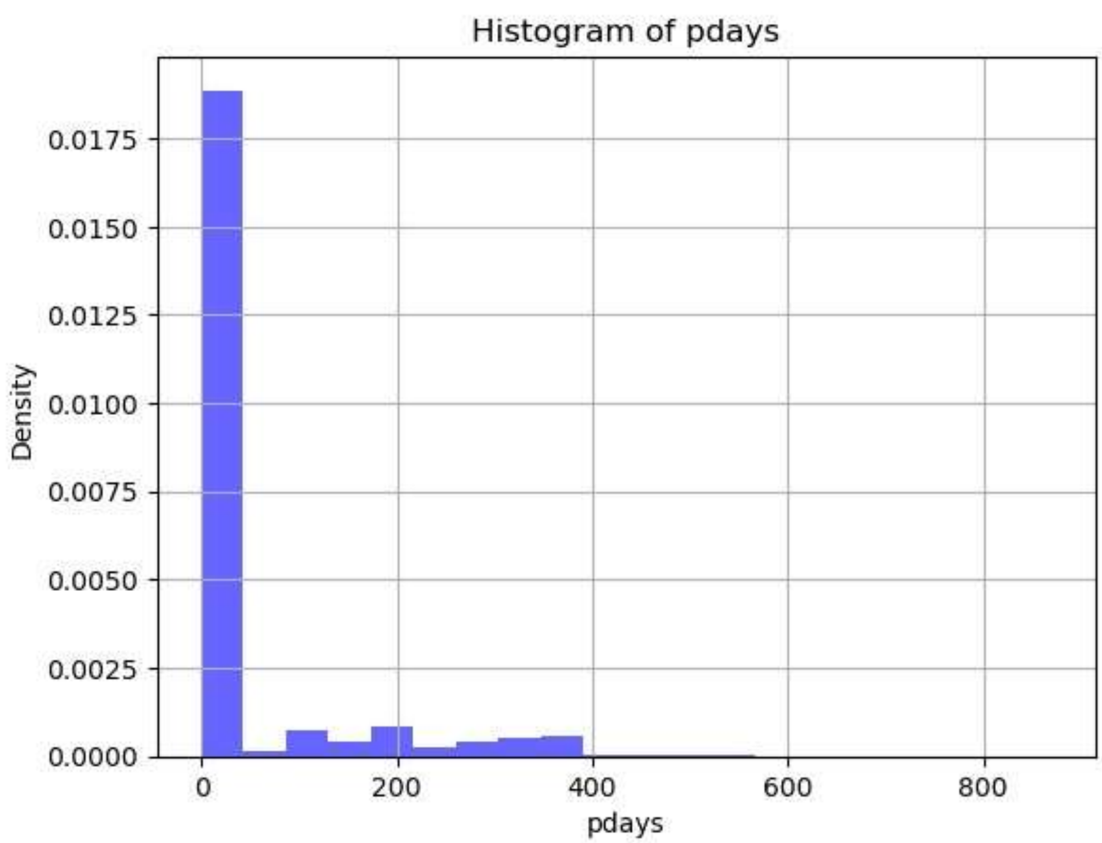

Boxplot of duration
1
0
1000
2000
3000
4000
5000

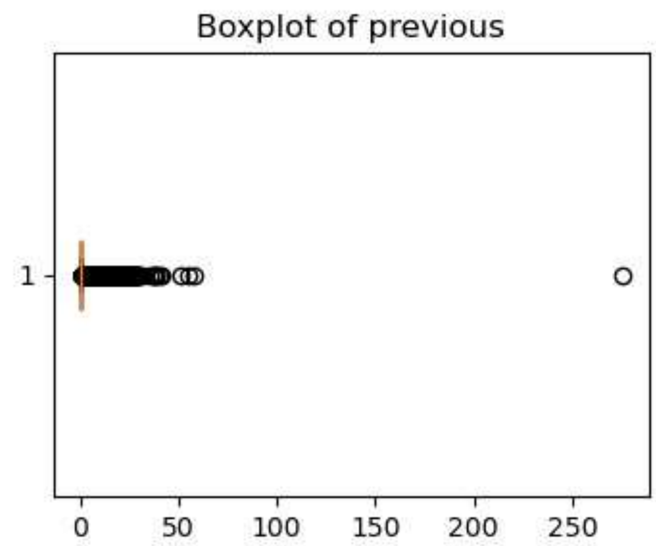

Histogram of duration
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Density
0
1000
2000
3000
4000
5000
duration

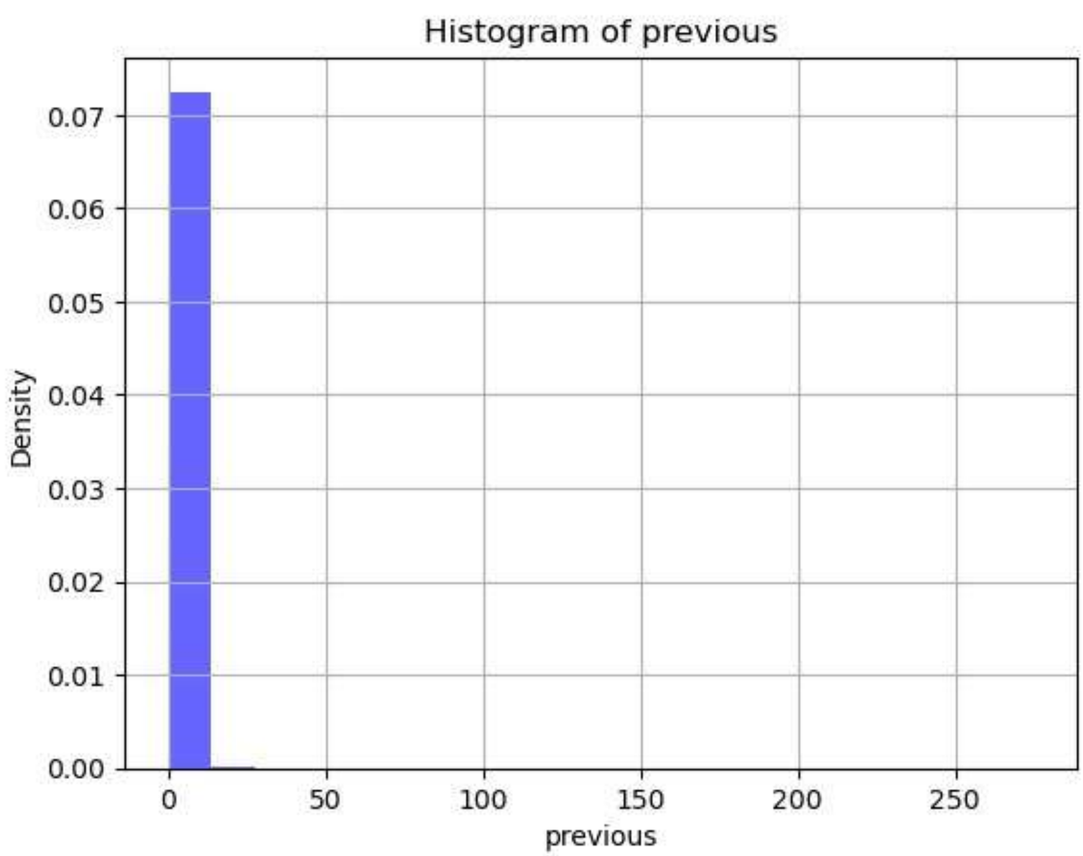

Boxplot of campaign
1
0
10
20
30
40
50
60


Histogram of campaign
Density
0.25
0.20
0.15
0.10
0.05
0.00
0
10
20
30
40
50
60
campaign


Boxplot of pdays
1
0
200
400
600
800


Histogram of pdays
Density
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
200
400
600
800
pdays





[28]:
```
df['marital'] =
df['marital'].map({'single':0,'married':1,'divorced':2})
df['education'] = df['education'].map({'secondary':0,'tertiary':1,
'primary':2}) df['default'] = df['default'].map({'yes':1,'no':0})
```

```
df['housing'] = df['housing'].map({'yes':1,'no':0}) df['loan'] =
df['loan'].map({'yes':1,'no':0}) df['contact'] =
df['contact'].map({'cellular':1,'telephone':0}) df['y'] =
df['y'].map({'yes':1,'no':0})
```

[29]: df

```
[29]:       age          job marital education default balance housing loan \
0       58   management 1      1      0      2143  1      0
1       44   technician 0      0      0      29     1      0 2   33
         entrepreneur    1      0      0      2      1      1
3       47   blue-collar        1      0      0      1506  1      0
4       33   blue-collar        0      0      0      1      0      0
 …    …            …      …       …       …       …        … …
45206   51   technician 1      1      0      825    0      0
45207   71   retired     2      2      0      1729  0      0
45208   72   retired     1      0      0      5715  0      0
45209   57   blue-collar        1      0      0      668    0      0
45210   37 entrepreneur 1      0      0      2971  0      0

       contact day month duration campaign pdays previous y
0      1      5      may    261    1      -1     0  0 1 1      5
 may   151    1      -1     0  0
2            1      5      may    76     1      -1     0  0
3            1 5 may 92 1 -1 0 0 4 1 5 may 198 1 -1 0 0 … … … … …
             … … … ..
45206        1 17  nov    977    3      -1     0  1
45207        1 17  nov    456    2      -1     0  1
45208        1 17  nov    1127  5      184    3  1
45209        0 17  nov    508    4      -1     0  0
45210        1 17  nov    361    2      188    11  0

      [45211 rows x 16 columns ]
```

[30]:
```
df=pd.get_dummies(df, columns=['job', "month"], prefix=["job", "month"],
    drop_first=True)
```

[31]: df

```
[31]:       age marital education default balance housing loan contact day \
 0      58     1      1      0      2143  1      0      1      5 1   44     0      0
  0     29     1      0      1      5 2    33     1      0      0      2      1      1
        1      5 3    47     1      0      0      1506  1      0      1      5
4 33 0 0 0 1 0 0 1 5 … … … … … … … … … …
45206   51    1      1      0      825    0      0      1      17
45207   71    2      2      0      1729  0      0      1      17
```

```
45208   72   1     0     0     5715  0     0     1     17
45209   57   1     0     0     668   0     0     0     17
45210   37   1     0     0     2971  0     0     1     17

     duration … month_dec month_feb month_jan month_jul month_jun \
0           261 …          0         0         0         0         0
1           151 …          0         0         0         0         0
2            76 …          0         0         0         0         0
3            92 …          0         0         0         0         0
4           198 …          0         0         0         0         0
…         … …        …        …        …        …        …

45206       977 …          0         0         0         0         0
45207       456 …          0         0         0         0         0
45208      1127 …          0         0         0         0         0
45209       508 …          0         0         0         0         0
45210       361 …          0         0         0         0         0

       month_mar month_may month_nov month_oct month_sep
0             0         1         0         0         0

1             0         1         0         0         0

2             0         1         0         0         0

3             0         1         0         0         0

4             0         1         0         0         0

…         …        …        …        …        …

45206         0         0         1         0         0

45207         0         0         1         0         0

45208         0         0         1         0         0

45209         0         0         1         0         0

45210         0         0         1         0         0

[45211 rows x 35 columns]
```

[32]:
```
df.shape
```

[32]: (45211, 35)

[33]:
```
df.info()
```

```
<class
'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to
```

```
45210 Data columns (total 35
columns):
 #  Column          Non-Null Count Dtype
--- ------           -------------- ---
                     --
0    age             45211     non-null
                     int64
1   marital          45211     non-null
                     int64
2   education        45211     non-null
                     int64
3   default          45211     non-null
                     int64
4   balance          45211     non-null
                     int64
5   housing          45211     non-null
                     int64
6    loan            45211     non-null
                     int64
7   contact          45211     non-null
                     int64
8    day             45211     non-null
                     int64
9   duration         45211     non-null
                     int64
 10 campaign         45211     non-null
                     int64
 11 pdays            45211     non-null
                     int64
 12 previous         45211     non-null
                     int64
 13 y                45211     non-null
                     int64
 14 job_blue-collar  45211     non-null
                     uint8
 15 job_entrepreneur 45211     non-null
                     uint8
 16 job_housemaid    45211     non-null
                     uint8
 17 job_management   45211     non-null
                     uint8
 18 job_retired      45211     non-null
                     uint8
 19 job_self-employed 45211 non-null uint8
 20 job_services     45211     non-null
                     uint8
 21 job_student      45211     non-null
                     uint8
```

```
 22 job_technician   45211      non-null
                     uint8
 23 job_unemployed   45211      non-null
                     uint8
 24 month_aug        45211      non-null
                     uint8
 25 month_dec        45211      non-null
                     uint8
 26 month_feb        45211      non-null
                     uint8
 27 month_jan        45211      non-null
                     uint8
 28 month_jul        45211      non-null
                     uint8
 29 month_jun        45211      non-null
                     uint8
 30 month_mar        45211      non-null
                     uint8
 31 month_may        45211      non-null
                     uint8
 32 month_nov        45211      non-null
                     uint8
 33 month_oct        45211      non-null
                     uint8
 34 month_sep        45211      non-null
                     uint8
dtypes: int64(14), uint8(21)
memory usage: 5.7 MB
```

[34]:
```
dependent_variable='y' independent_variables =
list(set(df.columns.tolist()) - {dependent_variable})
X =
df[independent_variables].copy()
y =
df[dependent_variable].copy()
```

[35]:
```
from sklearn.model_selection import train_test_split
```

[36]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.
↪2,random_state=101)
```

[37]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import CategoricalNB
from sklearn.cluster import KMeans
from sklearn.model_selection import cross_val_predict
from tabulate import tabulate
```

[38]:
```
knn = KNeighborsClassifier()
```

[39]:
```
knn.fit(x_train, y_train)
```

[39]: KNeighborsClassifier()

[40]:
```
knn_pred = knn.predict(x_test)
```

[41]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
```

[42]:
```
from sklearn.metrics import confusion_matrix,classification_report
```

[43]:
```
print(confusion_matrix(y_test,knn_pred))
```

```
[[7668  276]
 [ 803  296]]
```

[44]:
```
knn_report=classification_report(y_test,knn_pred)
print(knn_report)
```

```
              precision    recall  f1-score   support

           0       0.91      0.97      0.93      7944
           1       0.52      0.27      0.35      1099

    accuracy                           0.88      9043
   macro avg       0.71      0.62      0.64      9043
weighted avg       0.86      0.88      0.86      9043
```

[45]:
```
svm = SVC()
```

[46]:
```
svm.fit(x_train,y_train)
```

[46]: SVC()

[47]:
```
svm_pred = svm.predict(x_test)
```

[48]:
```
print(confusion_matrix(y_test,svm_pred))
```

```
[[7941    3]
 [1097    2]]
```

[49]:
```
svm_report = classification_report(y_test,svm_pred)
print(svm_report)
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94      7944
           1       0.40      0.00      0.00      1099

    accuracy                           0.88      9043
   macro avg       0.64      0.50      0.47      9043
weighted avg       0.82      0.88      0.82      9043
```

[50]:
```
from sklearn.linear_model import LogisticRegression
```

[51]:
```
model = LogisticRegression()
```

[52]:
```
model.fit(x_train,y_train)
```

```
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\linear_model\_log
istic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
    shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
                             https://scikit-
       learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[52]: LogisticRegression()

[53]:
```
pred = model.predict(x_test)
```

[54]:
```
print(confusion_matrix(y_test,pred))
```

```
[[7828 116]
 [ 890 209]]
```

[55]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      7944
           1       0.64      0.19      0.29      1099

    accuracy                           0.89      9043
   macro avg       0.77      0.59      0.62      9043
weighted           0.87      0.89      0.86      9043
avg
```

[56]:
```
kmeans = KMeans(n_clusters=2)
```

[57]:
```
kmeans.fit(x_train)
```

```
C:\Users\PHALGUN\anaconda3\lib\site-
packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
                                                          'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[57]: KMeans(n_clusters=2)

[58]:
```
train_cluster_labels = kmeans.predict(x_train)
test_cluster_labels = kmeans.predict(x_test)
```

[59]:
```
def map_cluster_labels(cluster_labels,
   true_labels): cluster_to_label = {} for
   cluster in set(cluster_labels): idx =
   cluster_labels == cluster label =
   true_labels[idx].mode()[0]
   cluster_to_label[cluster] = label
   mapped_labels = [cluster_to_label[cluster] for cluster in
   cluster_labels] return mapped_labels
```

[60]:
```
y_train_kmeans_mapped = map_cluster_labels(train_cluster_labels,
y_train) y_test_kmeans_mapped =
map_cluster_labels(test_cluster_labels, y_test)
```

[61]:
```
print(confusion_matrix(y_test,y_test_kmeans_mapped))
```

```
[[7944    0]
 [1099    0]]
```

[62]:
```
kmc_report=classification_report(y_test, y_test_kmeans_mapped)
print(kmc_report)
```

```
              precision    recall  f1-score   support

           0       0.88      1.00      0.94      7944
           1       0.00      0.00      0.00      1099
    accuracy0.88  9043 macro avg   0.44  0.50  0.47
 9043 weighted avg     0.77  0.88  0.82  9043
```

```
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\metrics\_classifi
cation.py:1344: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\metrics\_classifi
cation.py:1344: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\metrics\_classifi
cation.py:1344: UndefinedMetricWarning: Precision and F-score are
ill-defined and being set to 0.0 in labels with no predicted samples.
Use `zero_division` parameter to control this behavior.
_warn_prf(average, modifier, msg_start, len(result))
```

[63]:

```
models={
    "SVM":svm,
    "KNN":knn,
    "LOGISTICREGRESSION":model,
    "KMeans":kmeans,
}
```

[64]:

```
accuracy={}
precision={}
recall={}
f1={}
```

[65]:

```
for name, model in models.items():
    y_pred = cross_val_predict(model, x_test, y_test, cv=5)
    accuracy[name] = accuracy_score(y_test, y_pred)
    precision[name] = precision_score(y_test, y_pred, average='weighted')
    recall[name] = recall_score(y_test, y_pred, average='weighted')
    f1[name] = f1_score(y_test, y_pred, average='weighted')
```

```
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\linear_model\_log
istic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

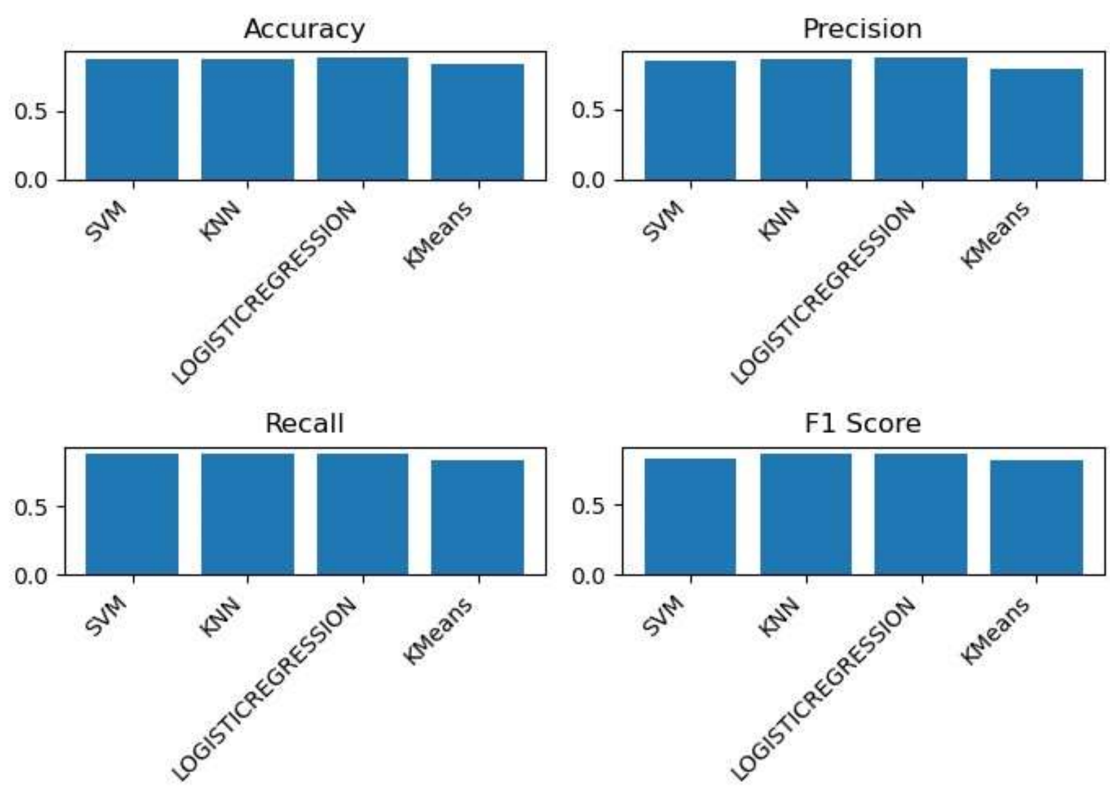
```
Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\linear_model\_log
istic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
    shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\linear_model\_log
istic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
    shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\linear_model\_log
istic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
    shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver
options:
    https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\PHALGUN\anaconda3\lib\sitepackages\sklearn\linear_model\_log
istic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
    shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-
    learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
  C:\Users\PHALGUN\anaconda3\lib\site-
  packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\PHALGUN\anaconda3\lib\site-
packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\PHALGUN\anaconda3\lib\site-
packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\PHALGUN\anaconda3\lib\site-
packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\PHALGUN\anaconda3\lib\site-
packages\sklearn\cluster\_kmeans.py:870:
```

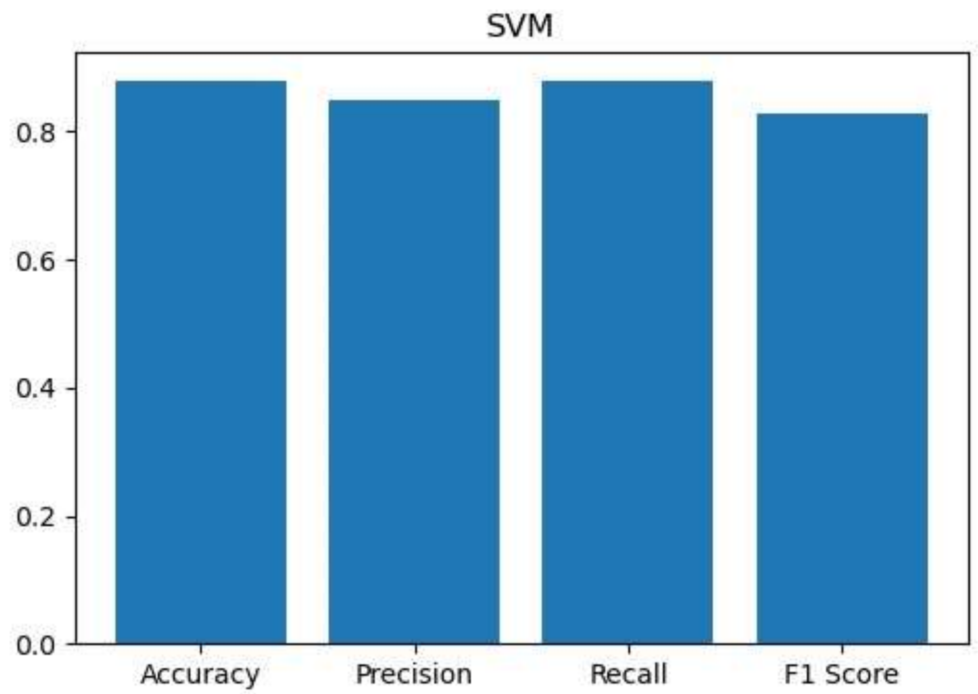
```
FutureWarning: The default value of `n_init` will change from 10 to
'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

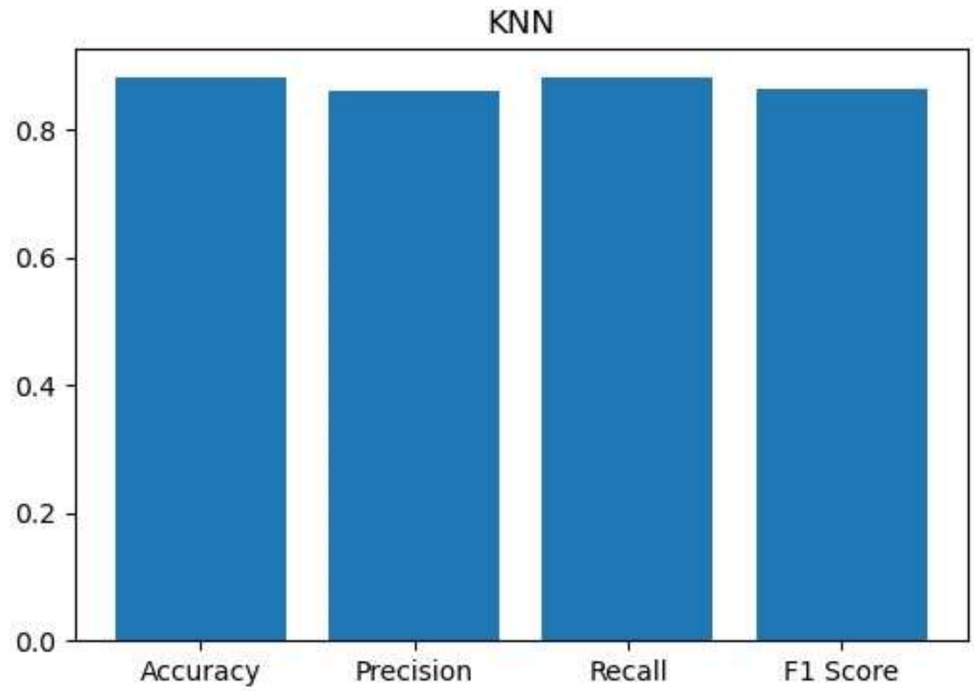
[66]:
```
fig, axes = plt.subplots(2, 2, figsize=(7,5))

axes[0, 0].bar(accuracy.keys(), accuracy.values())
axes[0, 0].set_title('Accuracy')
axes[0, 1].bar(precision.keys(), precision.values())
axes[0, 1].set_title('Precision')
axes[1, 0].bar(recall.keys(), recall.values())
axes[1, 0].set_title('Recall')
axes[1, 1].bar(f1.keys(), f1.values())
axes[1, 1].set_title('F1 Score')

for ax in axes.flat:
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

plt.tight_layout()
plt.show()
```

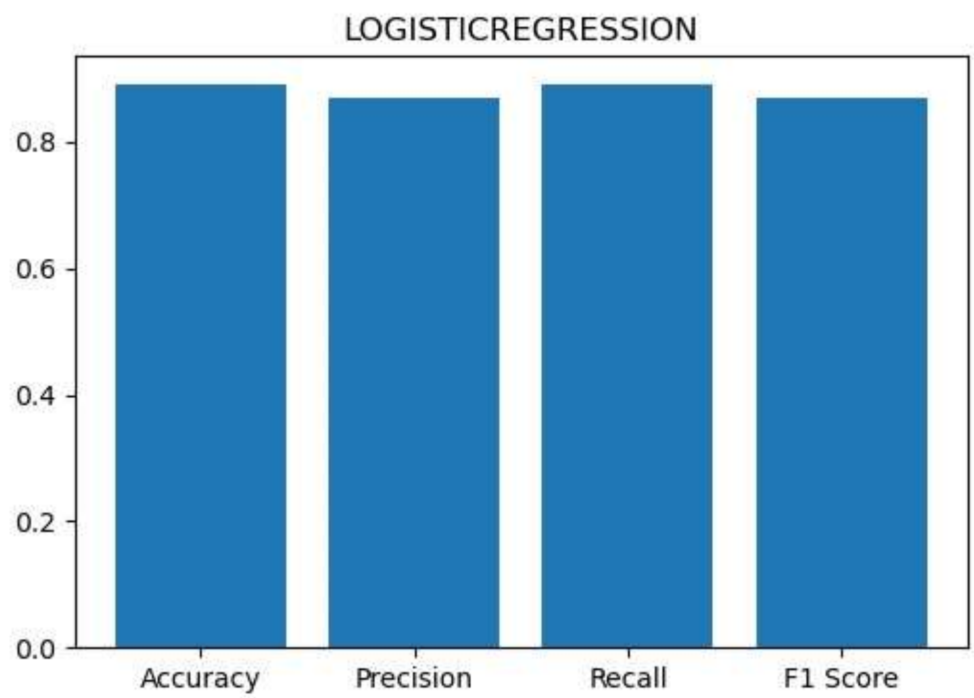
```
C:\Users\PHALGUN\AppData\Local\Temp\ipykernel_26316\785931337.py:13:
UserWarning: FixedFormatter should only be used together with
  FixedLocator ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
  ha='right')
```

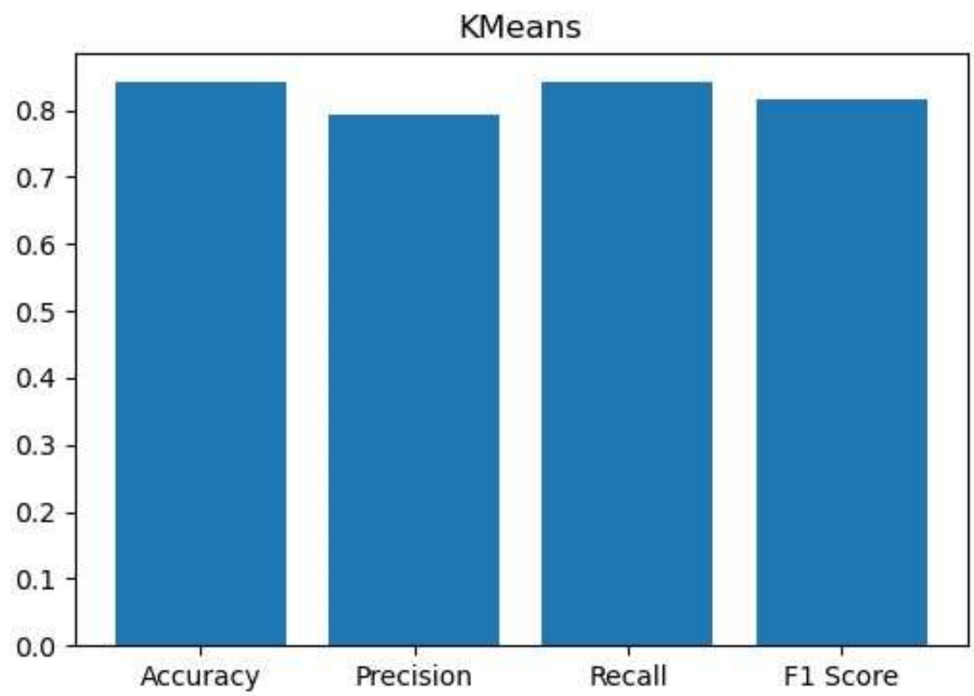


[67]:
```
for name in models.keys(): fig, axes
    = plt.subplots(figsize=(6,4))
          axes.bar(["Accuracy", "Precision", "Recall", "F1 Score"],
                                              [accuracy[name],␣
↪precision[name], recall[name],
    f1[name]]) axes.set_title(name)
    plt.show()
```




KNN
0.0
0.2
0.4
0.6
0.8
Accuracy
Precision
Recall
F1 Score


LOGISTICREGRESSION
0.0
0.2
0.4
0.6
0.8
Accuracy
Precision
Recall
F1 Score



[68]:
```
metrics_table = []
for name in models.keys():
    metrics_table.append({
        "Model": name,
        "Accuracy": accuracy[name],
        "Precision": precision[name],
        "Recall": recall[name],
        "F1 Score": f1[name]
    })
```

[69]:
```
print(tabulate(metrics_table, headers="keys", tablefmt="grid"))
```

```
+--------------------+-----------+-------------+----------+---------
---+
| Model              | Accuracy |   Precision | Recall  |  F1 Score |
+====================+===========+=============+==========+=========
===+
| SVM                | 0.879575 |   0.850356 | 0.879575 | 0.827157 |
+--------------------+-----------+-------------+----------+---------
---+
| KNN                | 0.881455 |   0.859081 | 0.881455 | 0.864535 |
+--------------------+-----------+-------------+----------+---------
---+
| LOGISTICREGRESSION | 0.890412 |  0.869769 | 0.890412 | 0.867771 |
+--------------------+-----------+-------------+----------+---------
---+
| KMeans             | 0.842088 |   0.793948 | 0.842088 | 0.814806 |
```

```
+--------------------+------------+-------------+----------+---------
---+
```

[ ]:

[ ]:

[ ]:

[ ]: